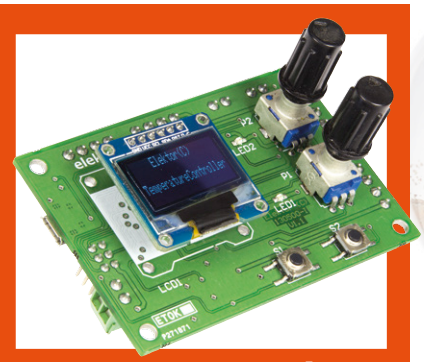
Januar/Februar 2017 | Nr. 553/554
www.elektormagazine.de

elektor

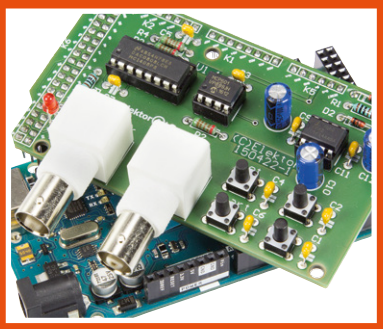
LEARN DESIGN SHARE

Sanduhr mit Arduino

Schreibt die Zeit in den Sand



Temperatur-Regler
Für 3D-Drucker und andere Projekte



Voltage-Tracker
Langzeitmessung für Oszilloskope

Schwebende Lampen (und andere Objekte)
Eine Anleitung

Ein ganz besonderes Modell!

In dieser Ausgabe:
9 Labor-Projekte
6 Leser-Projekte
3x Grundlagen
und vieles mehr!

Solider Arduino mit Kabelschraubklemmen • Internetradio mit Raspberry Pi • **LEDs in Farbe - gestern, heute, morgen** • ATtiny in Assembler programmieren • **Heizkesselmonitor mit ESP8266** • **SUPRA 2.0: Die MC-Version** • Eindraht-Sensoren • **Ampel für Modellbahn** • DAB+ Antennen-Diplexer • **Einfacher Temperaturwächter** • **Grundlagen: CIC-Filter** • Retronik: Elektor-Laborgeräte aus den 80ern und 90ern • **Web-Recherche: Bauen Sie einen Chipamp!** • Bemerkenswerte Bauteile: Quecksilber-Coulometer

G3078 E

(D) € 14,90 | CHF 27,00 | (A, B, L) € 16,00

4 198630 314904 02

Jetzt kostenloses Tagesticket sichern!
Geben Sie einfach folgenden Code unter
embedded-world.de/gutschein ein:
2ew17P

Nürnberg, Germany
14. – 16.3.2017

embeddedworld
Exhibition&Conference
... it´s a smarter world

Spüren Sie den Puls der Branche!

Die embedded world ist DER Treffpunkt
der internationalen Embedded-Community –
sichern Sie sich jetzt Ihren Wissensvorsprung!

embedded-world.de

Veranstalter Fachmesse
NürnbergMesse GmbH
T +49 9 11 86 06-49 12
besucherservice@nuernbergmesse.de

Veranstalter Kongresse
WEKA FACHMEDIEN GmbH
T +49 89 2 55 56-13 49
info@embedded-world.eu

Medien-partner
elektroniknet.de
computer-automation.de
ENERGIE & TECHNIK Solutions for a Smarter World
DESIGN & ELEKTRONIK KNOW-HOW FÜR ENTWICKLER
MEDIZIN-und-elektronik.DE
Elektronik
Elektronik automotive
Markt&Technik
Computer & AUTOMATION
MEDIZIN+elektronik

NÜRNBERG MESSE

Impressum

48. Jahrgang, Nr. 553/554
Januar/Februar 2017

Erscheinungsweise: 12 x jährlich
(Elektor Doppelheft im Wechsel mit dem Elektor Business Magazin)

Verlag
Elektor-Verlag GmbH
Kackertstraße 10
52072 Aachen
Tel. 02 41/955 09 190
Fax 02 41/955 09 013

Technische Fragen bitten wir per E-Mail an redaktion@elektor.de zu richten.

Hauptsitz des Verlags
Elektor International Media
Allee 1, NL-6141 AV Limbricht

Anzeigen:
Margriet Debeij (verantwortlich)
Tel. 02 41/955 09 174 / Fax 02 41/955 09 013
Mobil: +49 (170) 550 539 6
E-Mail: margriet.debeij@eimworld.com

Julia Grotenrath
Tel. 02 41/955 09 177 / Fax 02 41/955 09 013
Mobil: +31 6 510 530 39
E-Mail: julia.grotenrath@eimworld.com

Es gilt die Anzeigenpreisliste ab 01.01.2017.

Distribution:
IPS Pressevertrieb GmbH
Postfach 12 11, 53334 Meckenheim
Tel. 0 22 25/88 01-0
Fax 0 22 25/88 01-199

Der Herausgeber ist nicht verpflichtet, unverlangt eingesandte Manuskripte oder Geräte zurückzusenden. Auch wird für diese Gegenstände keine Haftung übernommen. Nimmt der Herausgeber einen Beitrag zur Veröffentlichung an, so erwirbt er gleichzeitig das Nachdruckrecht für alle ausländischen Ausgaben inklusive Lizenzen. Die in dieser Zeitschrift veröffentlichten Beiträge, insbesondere alle Aufsätze und Artikel sowie alle Entwürfe, Pläne, Zeichnungen einschließlich Platinen sind urheberrechtlich geschützt. Ihre auch teilweise Vervielfältigung und Verbreitung ist grundsätzlich nur mit vorheriger schriftlicher Zustimmung des Herausgebers gestattet. Die veröffentlichten Schaltungen können unter Patent- oder Gebrauchsmusterschutz stehen. Herstellen, Feilhalten, Inverkehrbringen und gewerblicher Gebrauch der Beiträge sind nur mit Zustimmung des Verlages und ggf. des Schutzrechtsinhabers zulässig. Nur der private Gebrauch ist frei. Bei den benutzten Warenbezeichnungen kann es sich um geschützte Warenzeichen handeln, die nur mit Zustimmung ihrer Inhaber warenzeichengemäß benutzt werden dürfen. Die geltenden gesetzlichen Bestimmungen hinsichtlich Bau, Erwerb und Betrieb von Sende- und Empfangseinrichtungen und der elektrischen Sicherheit sind unbedingt zu beachten. Eine Haftung des Herausgebers für die Richtigkeit und Brauchbarkeit der veröffentlichten Schaltungen und sonstigen Anordnungen sowie für die Richtigkeit des technischen Inhalts der veröffentlichten Aufsätze und sonstigen Beiträge ist ausgeschlossen.

© 2017 elektor international media b.v.
Druck: Senefelder Misset, Doetinchem (NL)
ISSN 0932-5468

Vielen Dank für Ihre Antworten!

Im letzten Editorial hatte ich Sie – liebe Leser – dazu aufgerufen, sich an unserer Leserbefragung zu beteiligen. Vor einigen Tagen habe ich mich bei den Kollegen erkundigt, wie viele Rückmeldungen wir erhalten haben, und war beeindruckt: International haben über 7000 Leser mitgemacht, 3500 alleine in Deutschland! Dafür möchte ich mich herzlich bei Ihnen bedanken. Die Ergebnisse werden wir im Newsletter am 23.12. veröffentlichen. Unter anderem hatten wir gefragt: Welche Themen möchten Sie noch ausführlicher in Elektor behandelt sehen? Und auch auf die Antwort, ob die Projekte noch genauer beschrieben werden sollen, waren wir alle sehr gespannt. Ich kann Ihnen die Antworten schon jetzt verraten: Von fast allen Themen, die wir vorgeschlagen hatten, wurde von einem bedeutenden Anteil unserer Leser „mehr“ gefordert. Und bei den Projekten wünschten sich über die Hälfte unserer Leser ausführlichere Beschreibungen.
So viel mehr an Inhalt passt in unsere bisherigen Hefte nicht mehr hinein. Wir haben uns daher entschlossen, Ihnen in diesem Jahr sowohl mehr gedruckte Artikel zu liefern als auch Veröffentlichungen, die wir besser im Internet aufgehoben sehen, dorthin zu verlagern. Damit bekommen Projekte und Grundlagen in den Heften mehr Platz.
Das altbekannte Elektor-Magazin wird ab 2017 sechs Mal im Jahr als „Doppelheft“ ausgeliefert – die 132 Seiten geben Raum für detaillierte Projekt-Beschreibungen und einen ausgewogenen Mix über viele Bereiche der Elektronik, der – wie wir aus dem Kiosk-Verkauf wissen - bei unseren Lesern besonders gut ankommt. Zusätzlich erhalten unsere Abo-Mitglieder sechs Mal im Jahr das Fachmagazin Elektor Business – es wird doppelt so dick sein wie bisher und redaktionell aufgewertet. Grundlagen-Artikel auf der Höhe der Zeit sowie Fachartikel zu neuen Technologien, die sich gewinnbringend im Entwickler-Alltag einsetzen lassen, haben hier ihren Platz. Tests und Vorstellungen neuer Produkte finden Sie ab jetzt auf unserer Website www.elektormagazine.de. Denn alles, was Sie ganz aktuell erhalten sollten, eignet sich am besten für diesen Kanal.
Auch wenn die Leserumfrage nun zu Ende ist: Uns Redakteure interessiert weiterhin sehr, wie unsere Artikel bei Ihnen ankommen. Natürlich auch in Elektor Business oder im Netz! Ich freue mich auf eine Mail von Ihnen unter redaktion@elektor.de.

Jens Nickel
Chefredakteur Elektor

Unser Team

Chefredakteur:	Jens Nickel (v.i.S.d.P.) (redaktion@elektor.de)
Ständige Mitarbeiter:	Dr. Thomas Scherer, Rolf Gerstendorf
Leserservice:	Ralf Schmiedel
Korrekturen:	Malte Fischer
Internationale Redaktion:	Thijs Beckers, Jan Buiting, Mariline Thiebaut-Brodier
Elektor-Labor:	Harry Baggen, Ton Giesberts, Luc Lemmens, Denis Meyer, Jan Visser, Clemens Valens
Grafik & Layout:	Giel Dols

Inhalt

48. Jahrgang – Nr. 553/554 Januar/Februar 2017

3 **Impressum**

6 **Das Elektor-Netzwerk**

8 **electronica Fast Forward Award 2016**
Die Gewinner

126 **Elektor World News**

128 **Katzdoku**
Sudoku für Elektroniker

LEARN DESIGN SHARE

10 **Willkommen bei LEARN**

11 **electronica 2016**
Neue Produkte im Überblick

14 **LEDs in Farbe**
Gestern, heute, morgen

20 **ATtiny in Assembler**
Kleine Mikrocontroller & Atmel Studio

26 **Grundlagen**
CIC-Filter

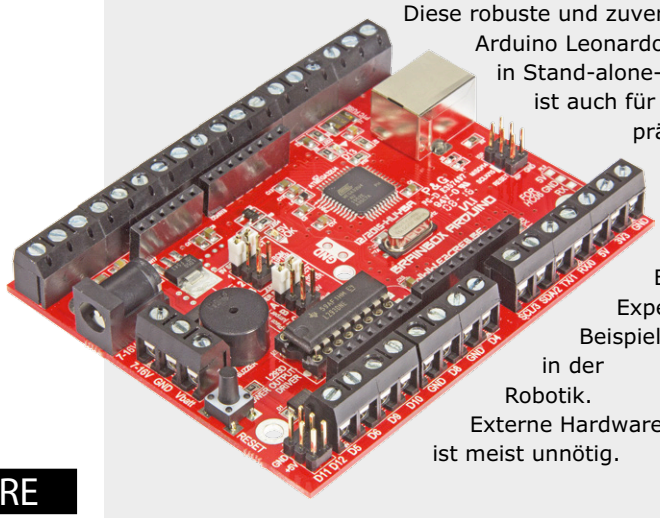
32 **Bemerkenswerte Bauteile**
Quecksilber-Coulometer

33 **Simulieren mit SystemVision**
Kostenlos und web-basiert

36 **Tipps und Tricks**
Von Lesern für Leser

BrainBox Arduino

Solider Arduino mit Kabelschraubklemmen



Diese robuste und zuverlässige Version des Arduino Leonardo wurde für den Einsatz in Stand-alone-Projekten entwickelt und ist auch für Lehr- und Lernzwecke prädestiniert. Der Onboard-Motortreiber-Baustein, die Kabelschraubklemmen und die wählbare Betriebsspannung machen Experimente einfach, zum Beispiel in der Robotik. Externe Hardware ist meist unnötig.

106

LEARN DESIGN SHARE

37 **Willkommen bei DESIGN**

38 **Sanduhr**
Schreibt die Zeit in den Sand

46 **Internetradio mit Raspberry Pi**
Schritt für Schritt zum eigenen Gerät

48 **Schwebende Lampen (und andere Objekte)**
Eine Experimentier-Anleitung

56 **Voltage-Tracker**
Langzeitmessungen für Oszilloskope

62 **Heizkesselmonitor mit ESP8266**
Haustechnik für die Energiewende

69 **Temperaturregler für 3D-Drucker**
Bringt auch Kaninchen warm über den Winter

74 **Sensoren (3)**
Für Arduino und Co.

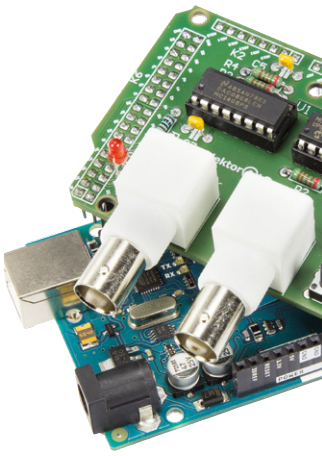
80 **MAXREFDES99# Display Driver Shield**
256 LEDs zu Ihrer Verfügung

82 **Swiss Pi – Flexibles Steuerungstool**
Software-Beispiele

90 **Verkehrssignal-Steuerung**
Für Straßenkreuzungen von Modellbahnanlagen

92 **Antennen-Diplexer**
Fürs Autoradio mit DAB+

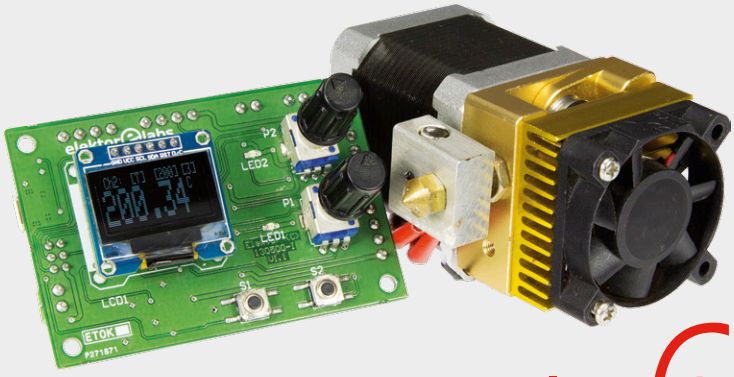
96 **Connected Objects mit Genuino 101**
Elektronik „funkt“ mit Mobilgerät



Sanduhr
Ein besonderes Modell!

38

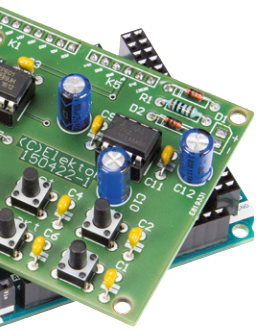
Schreibt die Uhrzeit in den Sand



Temperaturregler
Für 3D-Drucker und anderes

69

Jedes 3D-Druckfilament benötigt seine eigene Extrusionstemperatur, damit sich das beste Ergebnis erzielen lässt. Die optimale Temperatur kann selbst von der Farbe des Filaments abhängen! Deshalb ist ein exakter Temperaturregler für den Extruderkopf kein überflüssiger Luxus. Und wenn wir schon dabei sind, warum nicht auch die Mattentemperatur regeln?



102 Einfacher Temperaturwächter
Schaltet ein und aus

104 Klein, aber fein...
Elektor Breakout-Boards

106 BrainBox Arduino
Solider Arduino mit Kabelschraubklemmen

110 SUPRA 2.0: Die MC-Version
Signal ganz klein, Qualität ganz groß

114 HangTux
Hangman auf dem Raspberry Pi

elektor magazine

LEARN | DESIGN | SHARE

118 Willkommen bei SHARE

119 Retronik
Elektor-Laborgeräte aus den 1980ern und 1990ern

122 Web-Recherche
Bauen Sie einen Chipamp!

124 Aus dem Labor
Anschlüsse eines Opamps...

125 Elektormagazine.de/Labs
Projekte in der Pipeline

130 Projekt 2.0
Korrekturen, Updates und Leserbriefe

Vorschau

Das nächste Heft ist wie immer randvoll gefüllt mit Schaltungsprojekten, Grundlagen sowie Tipps und Tricks für Elektroniker.

Aus dem Inhalt:

- Uhr mit RGB-Ziffern
- Überblick Mikrocontroller-Starterkits
- OBD mit Raspberry Pi
- Sensoren am Arduino: Steuern mit Licht
- Supercaps
- Erweiterung für BBC micro:bit
- Professioneller Zufallszahlengenerator
- Leser-Projekt: Funkmodule und IoT-Gateway
- Musik-Instrument Theremin – mit FETs
- Linux-Board mit Modifikations-Möglichkeiten
- RFID in der Haus-Automatisierung
- Python für Mikrocontroller

Und vieles mehr!

Änderungen vorbehalten.
Elektor März/April 2017 erscheint am 22. Februar 2017.
Verkaufsstellen findet man unter www.pressekaufen.de.

Die Elektor-Community

LEARN | DESIGN | SHARE

Elektor durchbricht die Schranken einer Zeitschrift und wird zur Community aktiver E-Ingenieure - vom Anfänger bis zum Profi – begierig, überraschende Elektronik zu lernen, zu entwickeln, zu teilen.

Elektor-Shop: 24 Stunden an 7 Tagen der Woche für jeden Elektroniker geöffnet! Dauerhafter Rabatt von 10% für alle GOLD- und GREEN-Mitglieder. **www.elektor.de**

Elektor-Zeitschrift: 12 Ausgaben (6 Doppelhefte + 6 Business-Ausgaben) pro Jahr voll gepackt mit Elektronik-Projekten, Artikeln, Besprechungen, Tipps und Tricks. **www.elektormagazine.de**

Elektor-Platinen-Service: Bestellung von Platinen als Einzelstück oder Kleinserie. **www.elektorpcbservice.de**

Elektor wöchentlich & papierlos: Wöchentlicher digitaler Newsletter. Kostenlos und aktuell. **www.elektor.de/newsletter**

Elektor Academy: Webinare, Seminare, Präsentationen, Workshops, DVDs und mehr = praxisorientiertes Lernen. **www.elektor.de**

Elektor-Bücher & DVDs: Arduino, Raspberry Pi, Mikrocontroller und vieles andere mehr. Im Online-Shop mit 10% Rabatt für Mitglieder! **www.elektor.de/bucher**

Elektor.TV: Reviews, Eindrücke, Unboxings und persönliche Journale. Anschauen heißt Erfahrung sammeln. **www.elektor.tv**

Elektor-Labs: Eigene Projekte vorstellen – von Anderen lernen – Anderen helfen und mit Anderen teilen. Elektor macht mit und testet Ihre Ideen! **www.elektormagazine.de/labs**

Treten Sie dem weltweit größten Elektroniker-Netzwerk bei!

GREEN 1,78 €/Woche

- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 6x Elektor-Doppelheft (Digital)
- ✔ 6x Elektor Business (Digital)
- ✘ 6x Elektor-Doppelheft (Print)
- ✘ 6x Elektor Business (Print)
- ✔ Exklusive Top-Angebote
- ✘ Elektor Jahrgangs-DVD

Unsere Empfehlung

www.elektor.de/green-mitglied

GOLD 2,45 €/Woche

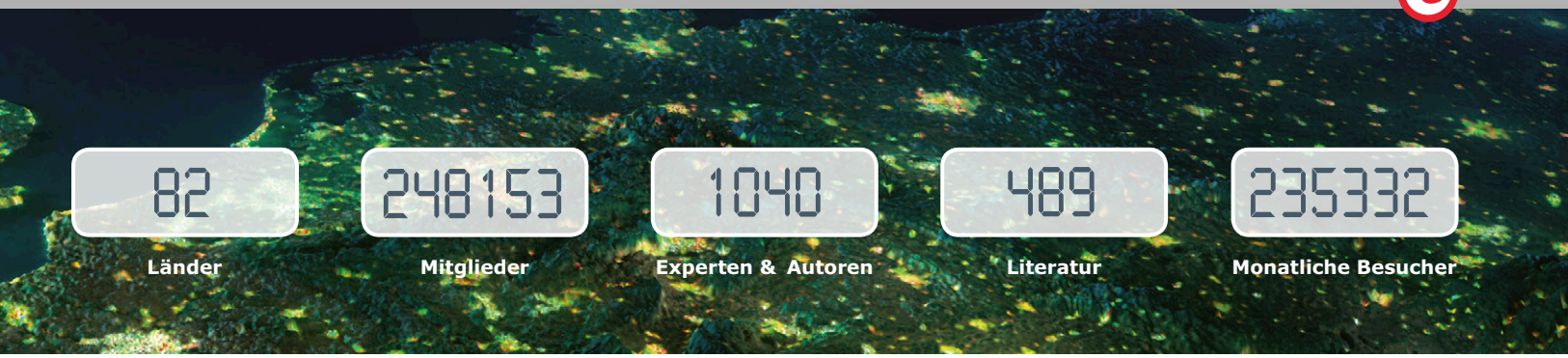
- ✔ Zugang zum Elektor-Archiv
- ✔ 10% Rabatt auf Shop-Produkte
- ✔ 6x Elektor-Doppelheft (Digital)
- ✔ 6x Elektor Business (Digital)
- ✔ 6x Elektor-Doppelheft (Print)
- ✔ 6x Elektor Business (Print)
- ✔ Exklusive Top-Angebote
- ✔ Elektor Jahrgangs-DVD

www.elektor.de/gold-mitglied

GRATIS

- ✘ Zugang zum Elektor-Archiv
- ✘ 10% Rabatt auf Shop-Produkte
- ✘ 6x Elektor-Doppelheft (Digital)
- ✘ 6x Elektor Business (Digital)
- ✘ 6x Elektor-Doppelheft (Print)
- ✘ 6x Elektor Business (Print)
- ✔ Exklusive Top-Angebote
- ✘ Elektor Jahrgangs-DVD

www.elektor.de/newsletter



www.elektormagazine.de

Eine ganze Welt mit Elektronik-News

Elektroniker entdecken eine ganze Welt mit Projekten, News, Videos und mehr auf unserer neu überarbeiteten Website. Rechts oben kann man einfach die Sprache auswählen. Zur Wahl stehen Deutsch, Englisch, Französisch und Niederländisch. Die intelligenten Such-Tools erleichtern das Auffinden von Artikeln und weiterführender Information. Registrieren Sie sich als GREEN- oder GOLD-Mitglied; mit Ihrem persönlichen Login haben Sie vollen Zugriff auf den Online-Shop inklusive vieler Extras. Hier können Sie auch Ihren Account überarbeiten – und das gilt auch für Ihre Mitgliedschaft, das gedruckte Heft und den Newsletter *Elektor*.

elektor: wöchentlich & papierlos

Ein wöchentlicher Newsletter vollgepackt mit Information

Mehr als 120.000 Elektroniker haben bereits ein Abonnement unseres kostenlosen *Elektor-Newsletters*. Jede Woche gibt es hier News, Tipps, Trends und mehr direkt in ihrem digitalen Briefkasten. Außerdem erhalten Sie so Zugriff auf weitere exklusive Projekte, Spezialangebote und Rabatte für den Online-Shop.
Registrieren Sie sich noch heute:
www.elektor.de/newsletter

electronica Fast

Wer hat 75.000 € Mediabudget und einen

Der „electronica Fast Forward Start-up Award powered by Elektor“, wie der Wettbewerb offiziell heißt, ist nicht nur für die drei Gewinner der großen Preise, sondern für alle Beteiligten ein großer Erfolg. Auf der electronica 2016 in München präsentierten 35 Teilnehmer aus 15 Ländern einer Jury aus Branchenkennern von STMicroelectronics, Conrad, Würth Elektronik und natürlich auch Elektor ihre Projekte. Nach drei Tagen der Präsentation all dieser hervorragenden Arbeiten wurden Rankings in den drei Kategorien „Idee“, „Prototyp“ und „Start-up“ erstellt. Am letzten Tag der Messe ging es für die ersten Preisträger der einzelnen Kategorien dann um den Hauptpreis, ein PR- und Marketingbudget im Wert von 75.000 € und einen Ausstellungsstand auf der electronica 2018.

Am von STMicroelectronics (Platin), Conrad (Gold), Würth Elektronik (Gold) und Trinamic (Bronze) gesponserten Wettbewerb nahmen Projekte teil, deren Bandbreite von der IoT-Security bis zu Monitoring-Systemen für ältere und körperlich

Die Preisträger

Idee

1. Artem Kuchukov (Deutschland) – Kewazo, gerüstbauender Roboter
2. Jonas Galle (Belgien) – Valcun, sicherer Low-cost-Desktop-Metall-3D-Drucker
3. Michael & Andrey Shustov (Russland) – Baristor, ein Barrierenwiderstand

Prototyp

1. Till Naumann & Lara Obst (Deutschland) – Mowea, modulare, netzunabhängige Windenergiesysteme
2. David Link & Christian Kind (Deutschland) – nevisQ, intelligente Sturzpävention und Sturzschutz
3. Peter Wasilewsky (Polen) – nWatch, tragbare MCU-Entwicklungsplattform

Start-up

1. JF Brandon (USA) – BotFactory Squink, Desktop-Platinendrucker
2. André Kholodov (Deutschland) – eCozy, intelligenter selbstlernender Thermostat
3. Milan Simek (Tschechien) – Sewio, präzises Objektverfolgungs- und Visualisierungssystem

Gesamtsieger

1. 75.000 € Mediabudget + Stand auf der electronica 2018: Mowea, modulare, netzunabhängige Windenergiesysteme
2. 50.000 € Mediabudget: BotFactory Squink, Desktop-Platinendrucker
3. 25.000 € Mediabudget: Kewazo, gerüstbauender Roboter

Spezialpreis „Tech for Good“

5.000 € Mediabudget:
Len Williams (Australien) – Every Drop Counts, digitales Wasserverbrauchs-Messsystem

Forward Award 2016

Stand auf der electronica 2018 gewonnen?

beeinträchtigte Menschen reichte, von modernsten 3D-Druckern zu RGB-Lampen, die von Pflanzen gesteuert werden, von Audiosystemen bis hin zu Präzisionsmessgeräten und Gerüstbau-Robotern. Konkurrenten aus Indien, Zentralrussland, den USA und sogar Australien, aber auch aus zahlreichen europäischen Ländern waren bereit, eine Woche in München zu verbringen, weil sie von dem, was sie geschaffen hatten, überzeugt waren. Wir fühlen uns geehrt und stolz, dass wir diese inspirierten Menschen und ihre Arbeit kennenlernen durften.

Weil es ja ein Wettkampf war, konnten wir leider den Großen Preis nicht an alle Teilnehmer vergeben. Aber allen wurde eine lebenslängliche Elektor-Helden-Mitgliedschaft verliehen und wir hoffen, von ihnen bald etwas über den Fortschritt ihrer Projekte zu hören.

Werden auch Sie beim nächsten Mal dabei sein?

(160278)

Die *modularen Windenergieturbinen* von Mowea wurden einstimmig zum Gesamtsieger des Fast Forward Award 2016 ausgerufen. Till Naumann (zweiter von rechts) erhielt den Grand Prize von electronica-Geschäftsführer Falk Senger (rechts), unterstützt von unseren Sponsoren (von links nach rechts) Shawn Silberhorn (Conrad), Alexander Gerfer (Würth Elektronik) und Jacky Perdrigeat (STMicroelectronics).

JF Brandon von BotFactory hatte sich auf den weiten Weg von Long Island City (USA) nach München gemacht, um seinen *Squink Desktop-PCB-Drucker* zu präsentieren. Es hat sich definitiv gelohnt, als er mit dem zweiten Preis nach Hause flog.

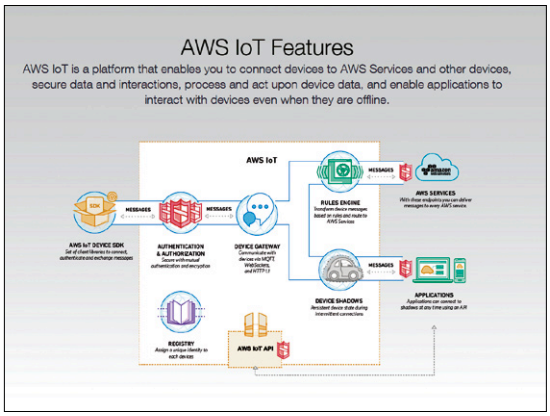
Artem Kuchukov von Kewazo (zweiter von links), offensichtlich sehr glücklich, erhielt den dritten Preis für das *Gerüstbau-Roboterprojekt*.

Willkommen bei LEARN

von **Jens Nickel**

Wenn einer eine Reise tut

Von Köln bis zur electronica muss man schon ein paar Stündchen Zug fahren. Und wie immer konnte ich weder auf ein funktionierendes Bahn-WLAN noch ein stabiles Handynetz zurückgreifen. Daher hatte ich ausgiebig Zeit, in mitgenommenen Elektronik-Fachmagazinen zu blättern. Und dort wurde ich mit vielen Artikeln zum IoT im Allgemeinen und zur IoT-Security im Besonderen schon einmal auf das Hauptthema der Münchner Messe eingestimmt. Diverse Boards und Kits von verschiedenen Herstellern, die dort beschrieben wurden, sind schon für die Zusammenarbeit mit einem der großen Cloud-Anbieter vorbereitet; und ein Name fällt hier besonders oft: Die Amazon Web Services des Cloud-Marktführers Amazon. Nach Anmeldung (limitierte Basisdienste sind kostenlos) kann man die Messwerte eigener Sensorboards in die Cloud hochladen, zum Beispiel über MQTT. Die Boards werden in der Cloud durch „Schattengeräte“ repräsentiert, welche die aktuellen Daten speichern. Auch wenn die Sensorboards offline sind, können Daten-Konsumenten wie Smartphones und Co. noch auf die Messwerte zugreifen.



Das Ganze muss ich mir unbedingt einmal näher ansehen, für meinen Blog „Mein Weg in das IoT“, den Sie unter www.elektormagazine.de/tags/mein-weg-ins-iot finden.

Zu viel zu sehen

Auf dem Elektor-Messestand gab es für mich jede Menge zu tun. Unter anderem wollte ich natürlich die Teilnehmer am Fast Forward Start-up Award einmal persönlich kennenlernen und die Prototypen genauer ansehen (ich habe es leider nicht ganz geschafft, es waren einfach zu viele interessante Sachen ausgestellt, auf dem Foto sieht man nur ein paar Beispiele).

(Zu) wenig Zeit blieb mir diesmal, auch die Stände der großen Halbleiterhersteller zu besuchen. Zum Glück hat mich unser freier Autor und Controller-Experte Viacheslav Gromov vertreten, seinen Messebericht finden Sie in diesem LEARN-Abschnitt.

Etwas wollte ich mir aber nicht entgehen lassen, nämlich den kombinierten Microchip/Atmel-Stand. Ich habe mir dort einen „Power“-Debugger zeigen lassen, außerdem auch ein Kit, das – Sie ahnen es – speziell für einen sicheren Zugang in das Internet of Things ausgelegt ist (www.atmel.com/tools/at88ckecc-aws-xstk.aspx). Hier ist ein Krypto-Chip von Atmel an Board, schon konfiguriert für eine Authentifizierung bei - tataaa - den Amazon Web Services. Ich muss gestehen, dass ich bei der detaillierten Erklärung durch einen Atmel-Experten etwas ins Schwimmen kam, aber das zu erkennen ist ja kein Fehler. Ich hatte es ja sowieso schon auf der Liste, dass ich mich mit (Root-) Zertifikaten, ECC und dem ganzen Anmelde- und Authentifizierungsprozess bei den AWS noch einmal näher beschäftigen muss… ⏮

(160244)

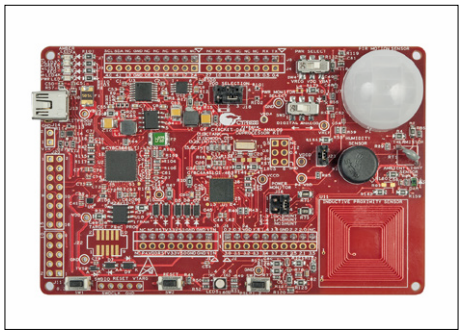
electronica 2016

Neue Produkte im Überblick

Von **Viacheslav Gromov** (D) (readers@gromov.de)

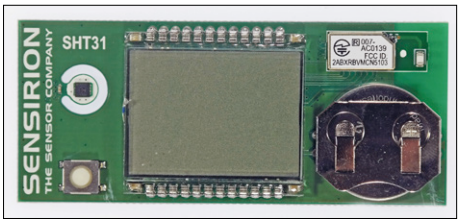
Mit 2.913 Ausstellern und rund 73.000 Besuchern aus 88 Ländern war die electronica 2016 so facettenreich wie noch nie. In allen Elektronik-Bereichen hat sich einiges getan: Von passiven Bauteilen und besonderen Steckverbindern über neue (Sicherheits-)MCUs bis hin zu neuen Sensoren und Bilderkennungssoftware reichte die Palette der Innovationen. Hier soll eine kleine Auswahl der vorgestellten Neuheiten präsentiert werden, die viele, aber weit nicht alle der genannten Bereiche abdeckt.

Gleich in zwei Bereichen konnte **Cypress** mit neuen Produkten punkten. Einerseits hat dieser Hersteller ein neues Mitglied seiner berühmten PSoC-MCU-Familie vorgestellt, nämlich den sogenannten PSoC Analog Coprocessor (48 MHz, 32 KB Flash). Diese 32-bit-MCU, die auf einem ARM-Cortex-M0+-Kern basiert, ist als Schnittstelle von der analogen zur digitalen Welt gedacht. An Bord sind ADCs, DACs, Operationsverstärker, Komparatoren, analoge Filter, Touch und vieles mehr. Um diese ganzen Möglichkeiten, die man dank der flexiblen PSoC-Technologie sehr gut ausnutzen kann, zu präsentieren, wurde das rund 49 $ teure CY8CKIT-048 PSoC Analog Coprocessor Pioneer Kit entwickelt: Dieses kleine Board verfügt über zahlreiche analoge Sensoren, die viele Anwendungen, von der Temperaturmessung über die induktive Abstands-Erkennung bis zum Bewegungsmelder abdecken [1].



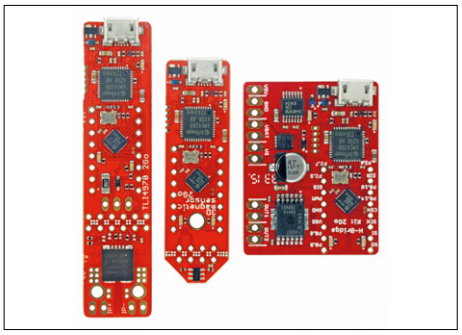
Der USB-C-Trend war auf der Messe deutlich zu spüren, bei Cypress hat sich natürlich auch einiges in diesem Bereich getan. Neu erschienen ist zum Beispiel der EZ-PD CCG3 (48 MHz, 64 KB Dual-Flash), der einen 32-bit-ARM-Cortex-M0-Kern hat und eine USB-C-Schnittstelle vollständig unterstützen kann, samt den ganzen inbegriffenen Schnittstellen und dem Stromversorgungs-Protokoll. Das schöne, rund 250 $ teure Board dazu heißt CY4531 EZ-PD CCG3 Evaluation Kit. Unabhängig davon ist auch der neue CY4500 EZ-PD Protocol Analyzer (Kosten: rund 200 $) ein sehr spannendes Produkt, das zum Beispiel zwischen einen Computer und die externe USB-C-Hardware geschaltet werden kann, um den Daten- und Stromfluss zu protokolieren und an der zusätzlichen USB-Micro-Schnittstelle auszugeben. Auf diese Weise kann man sowohl die programmierte Kommunikation zwischen zwei Geräten als auch die verwendeten Kabel und die Beschaltung testen [2].

Neben einigen raffinierten Gasflusssensoren hat **Sensirion** dieses Mal ein kleines aber feines SHT31 Smart Gadget Development Kit vorgestellt, welches schon für knapp



30 € zu haben ist und - wie der Name bereits verrät – den SHT31-Feuchtigkeits- und Temperatursensor ins Scheinwerferlicht rückt. Dieser neue, 2,5 x 2,5 mm² große Sensor wird über eine I²C-Schnittstelle mit der Außenwelt verbunden und misst in gewünschten Perioden die Temperatur und Feuchtigkeit (Messtoleranz: nur 2 % bzw. 0,3°C). Bei einer Messung pro Sekunde verbraucht dieser Sensor nur 2 µA. Das Board mit diesem IC ist aber nicht nur ein Gadget, bei dem man die Werte auf dem Display oder aufgrund von BLE auch auf einer Smartphone-App ablesen kann, es ist auch als ein Referenz-Design für Entwicklungen gedacht. Es verfügt über eine Knopfzellenhalterung, ein Display mit der 8-bit-MCU MC9S08LL8CGT als Treiber, einen Taster und das Nordic nRF51822 BLE-Modul (**B**luetooth **L**ow **E**nergy). Sowohl die Sensor-Pins als auch die Debug-Pins der MCU und des BLE-Moduls sind für Eigenentwicklungen auf der Platine gut zugänglich. Die ganzen Informationen zur Software und Hardware sind unter GitHub veröffentlicht, um mit Hilfe dieses Boards eigene Projekte umzusetzen [3].

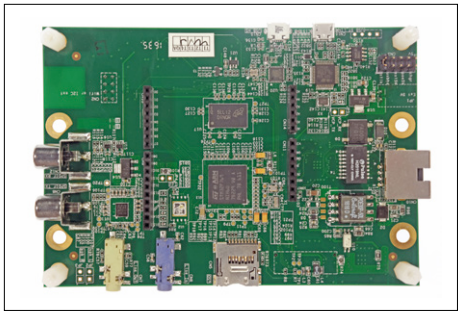
Infineon hat in allen Bereichen sehr viele Neuheiten herausgebracht. Neben unzähligen Arduino-Shields mit neuen ICs wurde auch die schon länger bekannte XMC 2Go Board-Familie mit dem XMC1100 (M0+, 48 MHz, 64 KB) als Basis erweitert. Hinzu kamen drei Ausführungen mit einem SPI-Strom-Sensor TLI4970 (Current Sensor 2Go), einem I²C-3D-Magnetfeld-Sensor TLV493D-A1B6 (3D Magnetic Sensor 2Go) und einem SPI-H-Brücken-IC IFX9201



(H-Bridge Kit 2Go). Gemeinsam haben die Boards außer der XMC-Basis, dass die wichtigen MCU-Ausgänge herausgeführt und jeweils zwei User-LEDs vorhanden sind. Dazu kommt die Besonderheit, dass man bei beiden Sensor-Boards die Sensoren von der Hauptplatine abtrennen kann, um sie zum Beispiel in eigenen Anwendungen zu verbauen. Für den Magnetfeld-Sensor gibt es noch zusätzlich eine Joystick-Erweiterung, die dann aus dem Sensor ein stufenloses Joystick-Bedienelement macht. Der Preis der einzelnen Boards beläuft sich im Schnitt auf rund 20 € [4].

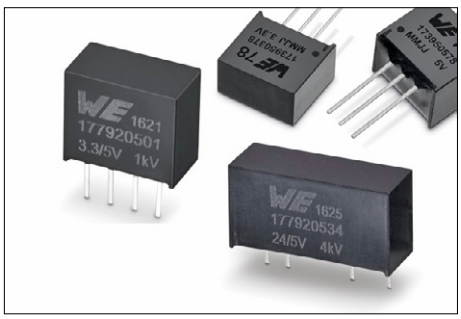
Das Thema Sicherheit war natürlich auf der electronica nicht unwichtig. Und Infineon hat dazu auch ein interessantes TPM-Produkt (**T**rusted **P**latform **M**odule, weltweites Sicherheits-Standard) namens SLB9670 vorgestellt. Dieses Sicherheits-IC, das per SPI angesteuert werden kann (es gibt auch den SLB9645 mit I²C), erhöht die Sicherheit eines Systems (welches eine einfache MCU oder ein ganzer PC sein kann), indem es bis zu acht 2048-bit-Schlüssel sicher abspeichern kann. Und diese Schlüssel verlassen nie das IC, alle nötigen Entschlüsselungen werden intern durchgeführt - der angeschlossene Prozessor wird also die Schlüssel im Betriebsablauf nie erfahren. Normalerweise werden solche Schlüssel für alle möglichen Verschlüsselungsarten auf einer SD-Karte oder anderweitigem Speicher abgelegt, wodurch sie natürlich vor Angriffen nicht sicher sind. Dieses IC lässt sich sehr gut zusammen mit dem Linux Kernel benutzen, eine interessante Beispiel-Anwendung ist damit der Raspberry Pi, für den extra ein Board entwickelt wurde, das bald erscheint. Dazu gibt es dann online reichlich Dokumentation, die zum Beispiel zeigt, wie man eine SSL-verschlüsselte Verbindung vom RPI zu einem PC sicherer macht [5].

STMicroelectronics erweitert die 32-bit-ARM-Cortex-M7-Familie der STM32-Sparte um mehrere Typen, wovon viele noch gar nicht in der Massen-Produktion sind. Es wurden aber bereits viele spannende MCUs und die dazu entwickelten Boards vorgestellt. Das aktuell neuste F7-Board, das bereits produziert wird, ist das STM32F769 Discovery Kit (Kosten: ca. 48 $) mit dem STM32F769NIH6 (216 MHz, 2 MB Flash), welcher vor allem durch seine Grafik-Leistung hervorsticht. Durch den eingebauten Chrom-ART Accelerator (DMA2D) und



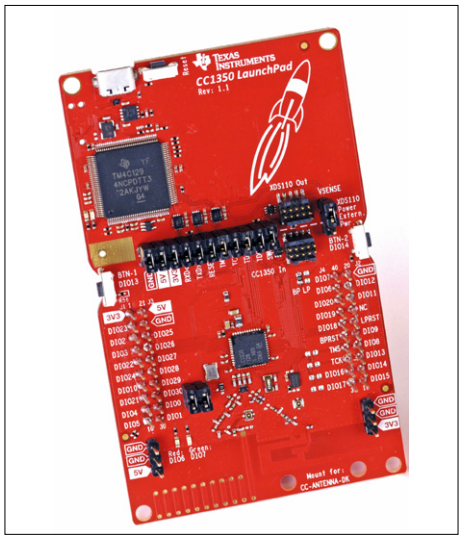
andere Peripherie beherrscht diese MCU eine Video-Ausgabe per DVI mit einer Auflösung von 720p und einer Bildfrequenz von 30 Hz. Das Besondere an diesem Board, das ansonsten alle typischen Eigenschaften eines normalen F7-ST-Discovery-Boards besitzt, ist die Erweiterungsmöglichkeit. Bei Bedarf kann man es mit einem TFT-Touch-Display-, einem DVI-zu-HDMI- oder einem WLAN-Adapter verbinden, falls das während der Entwicklungsphase benötigt wird [6].

Würth Elektronik hat weiter an seinen Power-Modulen gefeilt. Neu erschienen sind einige Vertreter der MagI³C-FISM (**F**ixed **I**solated **M**odule) und MagI³C-FDSM (**F**ixed Step **D**own Regulator **M**odule) Serien. Während MagI³C-FDSM ein Step-Down-Regler mit fester Ausgangsspannung (5 V oder 3,3 V mit max. 1 A) und einer hohen möglichen Eingangsspannung von bis zu 42 V ist (Wirkungsgrad bis zu 93 % möglich!), verfolgt der MagI³C-FISM eine ganz andere Strategie. Die Module dieser Serie sorgen für eine galvanische Trennung von



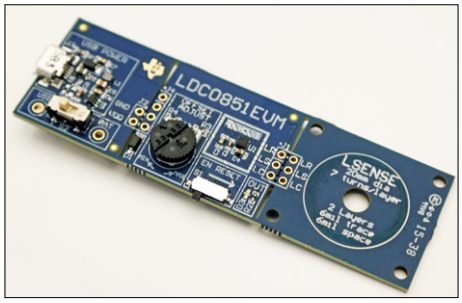
Eingang und Ausgang bis zu einer Spannung von 1 kV. Die Ausgangsspannung wird immer auf 5 V herabgesetzt, die Eingangsspannung kann bis zu 24 V betragen, die maximale Ausgangsleistung ist 1 W (Wirkungsgrad bis zu 80 % möglich!). Gemeinsame Merkmale der beiden Module sind die einfache Handhabung, die Sicherheit und die fast ganz wegfallende Beschaltung [7][8].

Auch **Texas Instruments** ließ mit neuen Produkten nicht lange auf sich warten. Viel Aufsehen erregt hat die neue Simple Link MCU CC1350. Diese verfügt außer dem 32-bit-ARM-Cortex-M3-Kern, der mit bis zu 48 MHz getaktet werden kann und dem ein Flash-Speicher von bis zu 128 KB zur Verfügung steht, über zwei Funk-Peripherieelemente. Einmal für die sogenannte Sub-1-GHz-Kommunikation und



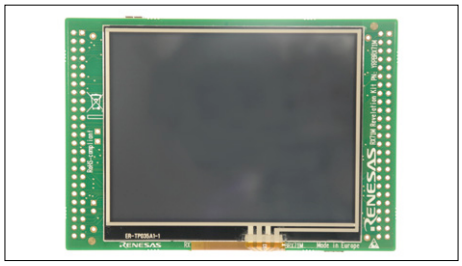
einmal für das weit verbreitete BLE. Der Vorteil dieser Kombination ist die intelligente Kommunikation innerhalb eines Netzwerks. Gerade im IoT- oder Industrie-4.0-Kontext wird ja eine übergreifende Vernetzung immer wichtiger. Mit dem CC1350 wird es möglich, dass beispielweise alle Sensoren und Aktoren in einem Haus via Sub-1-GHz-Funk (also genau genommen 868 MHz innerhalb der EU) über lange Distanzen untereinander kommunizieren und nur das zentrale Steuergerät auch gleichzeitig mit dem Smartphone des Endbenutzers per BLE verbunden ist. Zu diesem neuen Wireless-Mikrocontroller wurde selbstverständlich wie üblich auch ein LaunchPad entwickelt: das CC1350 LaunchPad, welches für 29 $ zu erwerben ist [9].

Zum ersten Mal spürte man auf der elec-

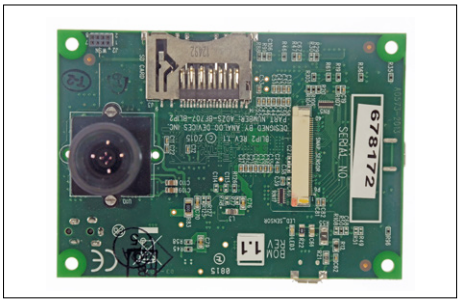


tronica einen Trend von der kapazitiven Touch-Technologie zur induktiven. Viele Hersteller bauten diese Technologie sogar schon in ihre MCUs ein. Der Vorteil ist vor allem die größere Beständigkeit gegen Umwelteinflüsse wie Wasser, Schmutz und so weiter. TI hat entsprechende, eigenständige kleine und große ICs entworfen, deren Einstellungen sehr einfach per Potentiometer vorgenommen werden können. Eines der kleineren Derivate ist der LDC0851, der mit sehr wenig externer Beschaltung und einer kleinen (auf der Platine gedruckten) Sensor-Spule auskommt. Eine kleine und mit 18 € günstige dazugehörige Entwicklungs-Platine heißt LDC0851EVM. Diese integriert einen einfachen induktiven Touch-Button, der vom USB-Micro-Anschluss oder der eingebauten Knopfzellenhalterung mit Strom versorgt wird. Die Platine kann man an zwei Bruchstellen brechen, um nur das IC oder nur die Sensor-Spule in eigenen Anwendungen benutzen zu können [10].

Renesas war mit zahlreichen Vorträgen sehr auf das neue RPBRX71M Revelation Kit (Kosten: 59 € netto) fokussiert, das die Vorteile des neuen Mitglieds RX71M der RX700-Familie vor allem anhand des verbauten 320x240-TFT-Touch-Displays praktisch vermittelt. Der RX71M ist eine 32-bit-MCU, die mit bis zu 240 MHz getaktet werden und bis zu 4 MB Flash beinhalten kann. Sie ist vor allem stark, wenn es um die Schnittstellen geht, aber auch in Hinblick auf die Sicherheit, da die MCU unter anderem zahlreiche Verschlüsselungsmöglichkeiten bietet sowie Kalibrierungs- und Test-Möglichkeiten für die interne Peripherie (z.B. ADC, GPIOs) [11].

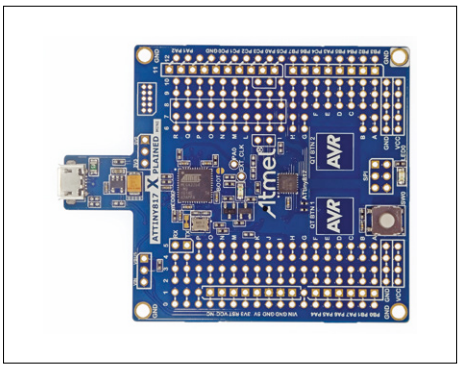


Außer den ganzen Neuheiten im analogen Bereich hat **Analog Devices** mit einigen Demos die Möglichkeiten des (nicht ganz neuen) ADZS-BF707-BLIP2 Boards bzw. der damit zusammenhängenden Technologie gezeigt. Dieses rund 200 Dollar teure Board mit zahlreichen Funktionen, vielen Schnittstellen und dem mit 400 MHz getakteten 32-bit-ADZS-BF707



ist mit seiner On-board-Kamera in der Lage, Gegenstände und sogar Gesichter zuverlässig zu erkennen. Um mit dem Board zu entwickeln, benötigt man den im Kit mitgelieferten ICE-1000-Emulator und das CrossCore Embedded Studio, was den Umgang mit dieser Technologie softwareseitig deutlich vereinfacht [12].

Und nun, um die weite **Atmel**- bzw. **Microchip**-Community mit dem Neuesten zu versorgen: Microchip hat stolz die neuen ATtinys 817, 816, 814 und 417 präsentiert. Damit wird die AVR-Sparte - wie versprochen (!) - nicht nur beibehalten, sondern stets weiterentwickelt.



Mit diesen vier neuen Mitgliedern ist nun die Anpassung an die größeren AVR- bzw. SAM-Familien fast vollständig gelungen. Denn diese vier 14- bis 24-Pin-MCUs mit 4 oder 8 KB Flash-Speicher verfügen außer den üblichen Peripherieelementen über einen echten, unabhängigen **P**eripheral **T**ouch **C**ontroller (PTC), ein **E**vent **S**ystem (ES), eine **R**eal **T**ime **C**lock (RTC) und mehrere benutzerprogrammierbare Logikblöcke (LUTs). Wie man an den Besonderheiten gut erkennen kann, werden die einzelnen Peripheriebereiche unabhängiger, was die CPU entlastet und ein noch besseres Echtzeitverhalten verspricht. Das Entwicklungsboard zu diesen neuen ATtinys ist ein vom Aufbau her schon gut bekanntes ATtiny817 Xplained Mini Board (Kosten: rund 12 $) [13].

Die PIC18F-Familie hat nun eine Sub-Familie mehr: die K40. Diese bewegt sich im Bereich von 28…64 Pins und 16…128 KB Flash sowie einer Taktfrequenz von bis zu 64 MHz. Highlights sind vor allem der **A**nalog to **D**igital **C**onverter with **C**omputation (ADC²), der auch ohne CPU-Beteiligung z.B. Mittelwerte berechnen oder Werte vergleichen kann (unter anderem auch für eine Touch-Anwendung nötig), und allgemein eine erhöhte Sicherheit. Diese wird erreicht durch einen Speicher CRC-Scan, einen besonderen Watchdog und einen sogenannten Hardware Limit Timer, der z.B. eingreift, wenn sich die angeschlossene externe Peripherie nicht rechtzeitig meldet. Viele der K40-Mitglieger sind in DIP-Gehäusen erhältlich, wodurch sich ein Einstieg mit dem bereits schon bekannten und sehr universellen Curiosity Development Board (Kosten: ungefähr 30 €) anbietet [14].

(160270)

Weblinks

[1] www.cypress.com/products/psoc-analog-coprocessor

[2] www.cypress.com/products/ez-pd-ccg3-type-c-port-controller-pd

[3] www.sensirion.com/de/produkte/feuchtesensoren/development-kit/

[4] http://goo.gl/o9hhGd

[5] http://goo.gl/Xu5Lkr

[6] www.st.com/en/evaluation-tools/32f769idiscovery.html

[7] http://katalog.we-online.de/de/pm/MAGIC-FISM

[8] http://katalog.we-online.de/de/pm/MAGIC-FDSM

[9] www.ti.com/tool/launchxl-cc1350

[10] www.ti.com/tool/ldc0851evm

[11] www.renesas.com/en-eu/solutions/key-technology/human-interface/rx71m-revelation.html

[12] http://goo.gl/LpdZig

[13] www.atmel.com/tools/ATTINY817-XMINI.aspx

[14] www.microchip.com/promo/pic18f67k40

LEDs in Farbe

Gestern, heute, morgen

Von **Dr. Thomas Scherer** (D)

Seit hundert Jahren kann die Menschheit anorganische Materie durch Strom zum Leuchten bringen. Und seit gut 50 Jahren gibt es sogar richtige LEDs. Diese farbigen Indikatorlämpchen ohne Glühdraht oder Gasfüllung wurden ganz langsam immer besser und vielfarbiger. In den letzten Jahren wurde der Wirkungsgrad so hoch, dass LED-Technik sogar die Leuchtstofflampe verdrängt. Die neueste Entwicklung: LEDs könnten sogar LCDs ersetzen.

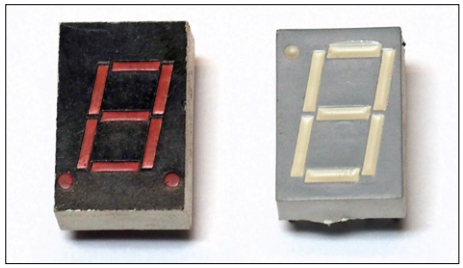


Bild 3. Aus der eigenen Antiquitätensammlung: Rot leuchtende Siebensegment-Displays mit einer Zeichenhöhe von 10 mm.

Bild 4. Die ersten Taschenrechner für den Massenmarkt waren mit winzigen roten LED-Siebensegment-Displays ausgestattet (Quelle: Erhaka, Wikimedia Commons).

Bild 5. LED-Versammlung. Obere Reihe von links nach rechts: Rote, gelbe und vier verschiedene grüne LEDs der 1970er Jahre im Kontrast mit einer modernen blauen LED, die so hell ist, dass der Bildsensor der Kamera übersteuert. Untere Reihe: Moderne grüne LEDs in 10 mm und 20 mm, RGB, weiß und infrarot.

Nach der Erfindung des Kristalldetektors im Jahre 1874 durch Ferdinand Braun dauerte es noch weitere 33 Jahre, bis Henry Joseph Round entdeckte, dass die Kontaktstelle zwischen Metallspitze und Kristall je nach Spannung gelblich, rötlich oder gar bläulich leuchten kann. Round war also der eigentliche Erfinder der LED und die von ihm entdeckte Elektrolumineszenz von Halbleitern wird ihm zu Ehren auch als Round-Effekt bezeichnet. Das war der Anfang.

Langsame LEDvolution

Die erste richtige LED stammte von RCA und leuchtete unsichtbar, da infrarot. Das war im Jahr 1955, der Entwickler hieß Rubin Braunstein. Texas Instruments gebührt die Ehre, ab 1962 die Serienproduktion von LEDs geschafft zu haben. Die infrarote TI-LED hatte immerhin schon einen Wirkungsgrad von 1,1 %! Die erste sichtbar rote LED wurde ebenfalls 1962 bei General Electric zum Leuchten gebracht. Nach einer erstaunlich langen Pause von zehn Jahren leuchtete dann erstmals eine LED in gelber Farbe – ebenfalls bei GE.

Rote LEDs wurden in den 1970ern schnell besser. Mit Wirkungsgraden bis zu 5 % wurden sie überall als Indikator eingesetzt. Die Exemplare im typischen runden 5- oder 3-mm-Plastikgehäuse brauchten mit rund 40 mW wenig Energie und boten weitere Vorteile: Sie waren anders als Lämpchen mit glühenden Wolframdrähten nicht erschütterungsempfindlich und hielten deutlich länger. Und Glimmlampen passten mit ihren Spannungen von 70...100 V doch eher ins gerade vergangene Röhrenzeitalter. LEDs sind mit ihrer Betriebsspannung von 1,2...4 V (je nach Farbe, siehe Kasten **Spannung & Farbe**) deutlich transistorkompatibler.

In den 1970ern gab es dann auch Anzeigen in LED-Technik in Form (roter) Sieben-Segment-Displays (**Bild 3**). Wer sich noch an die ersten Taschenrechner erinnern kann: Da waren winzige rote Displays verbaut (**Bild 4**), die in der Regel mehr Strom verbrauchten als der zum Rechnen befähigte Chip. Mein TI-59 von 1977 – einer der ersten programmierbaren Taschenrechner – konnte zwar entschieden mehr, hatte aber ein ähnlich mickriges Display. Und trotzdem musste ich als Schüler lange darauf sparen ;-).

In den 1970ern waren mit roten, gelben und grünen LEDs immerhin schon drei Farben verfügbar. In der oberen Reihe von **Bild 5** sieht man (mit Ausnahme der blauen LED rechts) originale „antike“ LEDs aus den 1970ern. Sie waren oft nicht besonders hell, Rot- und Gelbtöne variierten und grün war auch nicht unbedingt grün. **Bild 6** zeigt reale Helligkeits- und Farbunterschiede grüner LEDs aus dieser Zeit. Alle vier LEDs sind in Serie geschaltet und von einem Strom von 5 mA durchflossen. Nur die LED links

Spannung & Farbe

Anorganische LEDs basieren auf dotierten Halbleitern, wobei die Elemente Aluminium, Arsen, Gallium, Indium, Phosphor und Stickstoff häufig vorkommen. Neuerdings wird auch an LEDs mit Kohlenstoff bzw. Diamant, Silizium und Zink geforscht. Zusammensetzung, Konzentration und Aufbau haben nicht nur Auswirkungen auf den Wirkungsgrad, sondern vor allem direkt auf die Farbe des abgegebenen Lichts. Die Energie der emittierten Photonen hängt nämlich direkt von der sogenannten Bandlücke ab, der Energiedifferenz zwischen Leitungs- und Valenzband der verwendeten Halbleiter. Je größer die Bandlücke, desto höher die Photonenenergie und desto kurzwelliger das Licht. Die Bandlücke hat großen Einfluss auf die Schwellenspannung, die es zu überwinden gilt, damit a) Strom fließt und b) die Elektronen beim Übergang von Leitungs- zu Valenzband ein Photon emittieren. Der prinzipielle Zusammenhang ist in **Bild 1** wiedergegeben. Kein Wunder also, dass die Betriebsspannung von blauen LEDs etwa doppelt so hoch ist wie die von roten LEDs, denn die Lichtfrequenz ist proportional zur Photonenenergie.

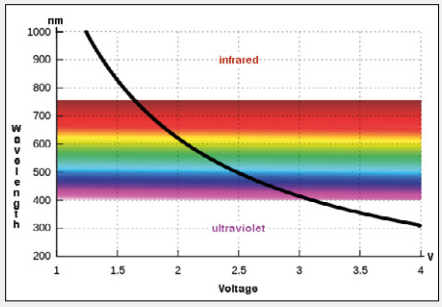


Bild 1. Der Zusammenhang zwischen der Betriebsspannung (Bandlücke) und der Lichtfarbe von LEDs. Spannung und Wellenlänge sind umgekehrt proportional.

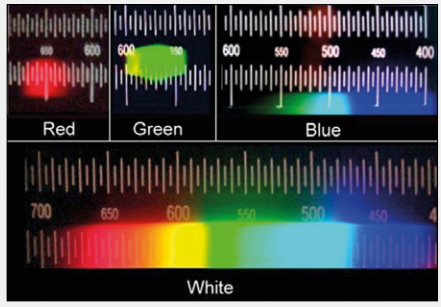


Bild 2. Spektren verschiedenfarbiger LEDs. Je kürzer die Wellenlänge, desto breitbandiger das Spektrum. Unten zum Vergleich das Spektrum der Sonne (Quelle: Wikimedia Commons).

Tabelle 1 zeigt, für welche Farbe bzw. welchen Wellenlängenbereich welche Halbleiter zum Einsatz kommen und mit welchen Betriebsspannungen man dabei rechnen muss. Übrigens hängt die reale, an den LED-Klemmen gemessene Spannung selbstverständlich nicht nur von der Bandlücke des Materials, sondern neben sonstigen Feinheiten auch von den Material- und Bonding-Widerständen sowie Materialübergängen ab. Mit steigendem Strom steigt also auch die an einer LED abfallende Spannung. Dass es keinen scharfen Übergang zwischen Stromleitung und Nichtleitung gibt, liegt daran, dass gerade im Übergangsbereich die statistische Wahrscheinlichkeit, dass ein Elektron die Bandlücke „überwindet“, mit steigender Spannung steigt. Übrigens verschiebt sich das Spektrum von LEDs mit steigender Temperatur leicht in Richtung größerer Wellenlängen, also zu wärmeren Farben. Auch das ist kein Wunder, denn mit steigender Temperatur sinkt die Betriebsspannung. Wenn man für bestimmte Anwendungen auf stabile Lichtfarben angewiesen ist, muss man die Chip-Temperatur der LEDs im Auge behalten.

LED-Licht liegt bezüglich Reinheit irgendwo zwischen (farbigen) Glühlampen und Lasern. Mit anderen Worten: LEDs sind nicht so schmalbandig wie Laser, die fast nur Licht einer Wellenlänge emittieren, aber auch lange nicht so breitbandig wie Glühlampen. **Bild 2** zeigt, dass besonders im Bereich größerer Wellenlängen LEDs recht schmalbandig sind. Das Spektrum blauer LEDs hingegen ist deutlich breiter. Letzteres kommt weißen LEDs zugute, die ja aus blauen LEDs plus einer Phosphorschicht bestehen. Ein gleichmäßigeres Spektrum hat ein „natürlicheres“ bzw. sonnenähnlicheres Licht zur Folge. Die damit zusammenhängende Lichtqualität weißer LEDs wird durch den Farbwiedergabeindex Ra ausgedrückt. Die Sonne kommt auf den Referenzwert von 100 und weiße LEDs erreichen Werte zwischen 75 und 95.

Tabelle 1

Farbe	Wellenlänge (nm)	Material		Spannung (v)
infrarot	>760	GaAs AlGaAs	(Galliumarsenid) (Aluminiumgalliumarsenid)	<1,6
rot	610 – 760	AlGaAs GaAsP AlGaInP GaP	(Aluminiumgalliumarsenid) (Galliumarsenidphosphid) (Aluminiumgalliumindiumphosphid) (Galliumphosphid)	1,6 – 1,9
orange	590 – 610	GaAsP AlGaInP GaP	(Galliumarsenidphosphid) (Aluminiumgalliumindiumphosphid) (Galliumphosphid)	1,8 – 2,2
gelb	570 – 590	GaAsP AlGaInP GaP	(Galliumarsenidphosphid) (Aluminiumgalliumindiumphosphid) (Galliumphosphid)	2,0 – 2,4
grün	500 – 570	InGaN GaN GaP AlGaInP AlGaP	(Indiumgalliumnitrid) (Galliumnitrid) (Galliumphosphid) (Aluminiumgalliumindiumphosphid) (Aluminiumgalliumphosphid)	2,2 – 2,7
blau	450 – 500	ZnSe InGaN SiC	(Zinkselenid) (Indiumgalliumnitrid) (Siliziumkarbid)	2,6 – 3,3
violett	400 – 450	InGaN	(Indiumgalliumnitrid)	3,2 – 3,6
ultraviolett	230 – 400	AlN AlGaN AlGaInN	(Aluminiumnitrid) (Aluminiumgalliumnitrid) (Aluminiumgalliumindiumnitrid)	3,5 – 4,2

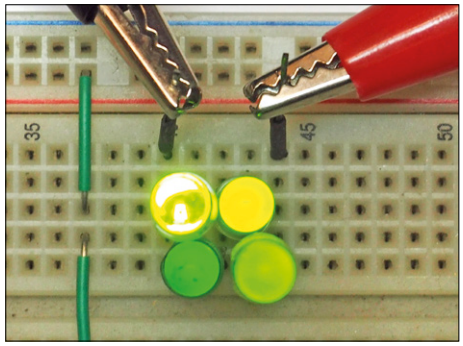


Bild 6. Vier unterschiedliche grüne LEDs: Jede leuchtet anders. Die LED links oben ist ein superhelles, modernes Exemplar, das die Kamera übersteuert. Die anderen drei 5-mm-LEDs sind die Exemplare von Bild 5.

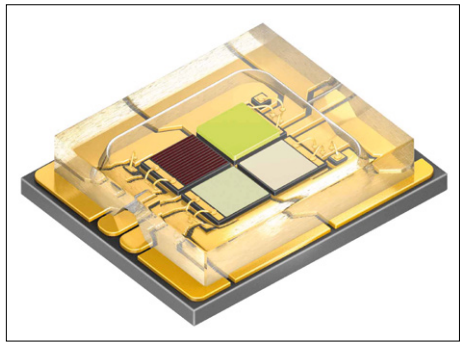


Bild 9. Mehr-als-RGB-LED: Im SMD-Gehäuse der OSTAR-LED hat OSRAM neben den Primärfarben noch einen weißen LED-Chip untergebracht.

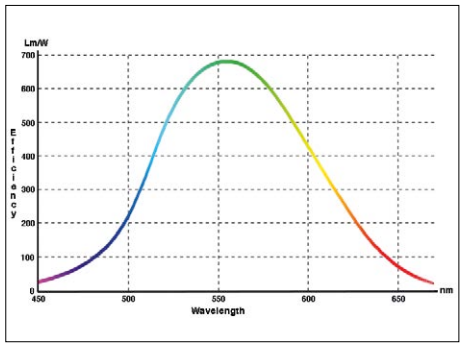


Bild 7. Maximale theoretische Lichtausbeute von LEDs in Abhängigkeit von der Farbe. Achtung: Die Einheit Lumen gibt einen von der Farbe abhängigen Helligkeitseindruck des menschlichen Auges an.

Bild 8. LED-Kerze in „schön". Bei dieser E14-Lampe wurden die Glühdrähte mit LED-Streifen nachgebildet (Quelle: OSRAM).

oben ist ein neues High-Efficiency-Exemplar zum Vergleich. Sie ist in Wirklichkeit satt grün, aber so hell, dass dies den Kamerasensor in die Sättigung treibt. Eine der Oldie-LEDs ist eher gelbgrün als grün. Bei roten und gelben LEDs war das damals nicht anders. Aber moderne LEDs im 5-mm- oder SMD-Gehäuse haben so klare Farben, dass es eine Freude ist. Bei der Verwendung als Indikator muss man mit dem Strom aufpassen, denn sie sind so hell, dass sie leicht blenden.

Status quo

Es hat länger gedauert, bis die fehlende Farbe Blau hinzukam: Erst in den 1990ern brachte Nichia blaue LEDs auf den Markt. Die blaue LED war übrigens einen Nobelpreis wert, den die Entwickler Shuji Nakamura, Samu Akasaki und Hiroshi Amano im Jahre 2014 überreicht bekamen – eine große Ehre für ein kleines Bauteil. Der Preis war berechtigt, denn hocheffiziente blaue LEDs bilden die Basis für weiße LEDs. Bei diesen wurde lediglich noch eine Phosphorschicht aufgebracht, die einen Teil des hochfrequenteren blauen Lichts in gelbliches Licht umwandelt, was dann zusammen mit dem restlichen blauen Licht durch additive Farbmischung weißes Licht ergibt. Dieses Verfahren wurde 1995 von Jürgen Schneider 1995 am Fraunhofer Institut entwickelt und bildet heute die Basis fast aller LEDs, die für Beleuchtungszwecke genutzt werden. Weiße LEDs haben heute einen knapp höheren Marktanteil als farbige. Mehr über weiße LEDs zu Lichtzwecken ist in meinem Artikel „Es werde LED!" in Elektor Januar 2016 [1] zu lesen.

In Bild 5 rechts oben sieht man eine blaue LED im klaren Kunststoffgehäuse, die mit 5 mA betrieben wird. Sie ist so hell, dass auch sie den Kamerasensor übersteuert und so fälschlicherweise hellbau-weißlich wirkt. Links unten sieht man je eine grüne LED mit 10 mm und 20 mm Gehäusedurchmesser. Das dickere Exemplar ist eine Multi-Chip-LED mit vier in Serie geschalteten grünen LED-Chips. Rechts daneben sieht man eine RGB-LED mit gemeinsamer Anode, daher die vier Beinchen. Durch entsprechende Ansteuerung der drei in den Primärfarben gehaltenen integrierten Chips kann man damit eine fast beliebige Mischfarbe erzeugen. Die klare LED rechts daneben ist ein modernes weißes Exemplar. Ganz rechts schließlich befindet sich eine infrarote LED. Ihre Gehäusefärbung lässt das IR-Licht problemlos passieren.

Die zunächst mauen Lichtausbeuten der Siebziger steigerten sich am Anfang dieses Jahrtausends schnell auf Effizienz-Werte von bis zu 100 Lm/W. Damit waren weiße LEDs auf einem Niveau, wo sie Leuchtstofflampen Konkurrenz machen konnten, da sie bei ähnlicher Lichtausbeute kein Quecksilberproblem haben und auch noch länger halten. Mittlerweile erreichen weiße LEDs im Labor eine Lichtausbeute, die mit >300 Lm/W nur noch wenige Prozent unter dem theoretischen Limit (350 Lm/W bei 6.600 K) liegt. **Bild 7** zeigt die maximale theoretische Lichtausbeute in Abhängigkeit von der Lichtfarbe. Dabei ergibt sich sogar ein schönes Pseudoparadoxon: Obwohl weiße LEDs auf blauen Chips basieren, die nicht über 200 Lm/W hinauskommen, schaffen weiße LEDs mehr. Der Grund hierfür ist, dass die Einheit Lumen die spektrale Empfindlichkeit des menschlichen Auges berücksichtigt, und wir sehen eben grünes Licht bei gleicher Leistung sehr viel heller. Obwohl also bei der Umwandlung in gelbliches Licht im Phosphor ein Teil der Energie verloren geht (in Wärme umgewandelt wird), erscheint uns die weiße LED heller als der rein blaue Chip unter dem Phosphor.

Und noch etwas ist interessant: Da LEDs zumindest im Labor weitgehend ausgereizt sind, wird es nur noch minimale Steigerungen beim Wirkungsgrad geben. Der technische Fortschritt wird sich nur noch in den fertigen Produkten zeigen. Bei LEDs in Massenproduktion ist also durchaus noch Luft nach oben, bis sie die fantastischen Lichtausbeuten der Laborexemplare zu bezahlbaren Preisen erreichen. Dieser Prozess dürfte in weni-

ger als einem Jahrzehnt abgeschlossen sein, und dann steht der Menschheit dank modernem Licht aus Halbleitern durchweg eine besonders energiesparende Beleuchtungstechnik zur Verfügung, die aufgrund der langen Lebensdauer auch noch einen relativ geringen ökologischen Fußabdruck mit sich bringt.

Farbige LEDs

In den letzten Jahren wurde der Wirkungsgrad von LEDs so hoch und die Preise dank Massenproduktion so niedrig, dass sich Lampen auf LED-Basis auf breiter Front durchsetzen. Und es gibt nicht nur die manchmal recht ungelenk aussehenden LED-Lampen, die intern mit vielen SMD-Chips bestückt sind, sondern es werden auch die Filamente schwacher kerzenförmiger Glühbirnen mit LED-Strukturen nachgebildet (**Bild 8**), sodass auch Kronleuchter fast stilecht aber modern und energiesparend bestückt werden können. Es gibt mittlerweile Lampen in fast allen Formen – und Farben – womit wir wieder beim Thema wären.

Interessanterweise werden Infrarot-LEDs nicht nur seit Jahrzehnten bei Fernbedienungen oder heute als unauffällige Nachtbeleuchtung bei Überwachungskameras oder für militärische Zwecke eingesetzt, sondern IR-Lampen mit hunderten IR-SMD-LEDs und Anschlussleistungen über 100 W dienen mittlerweile in der Industrie als Basis für Projektoren, die einen bestimmten Abschnitt einer Produktionsstraße oder eine Montagestation mit einem infraroten Lichtgitter ausleuchten. Zweck der Übung ist: Im Gegensatz zu Menschen können die Kameras von Robotern diese Linien sehen und sich so nicht nur im Raum gut orientieren, sondern auch die Lage von Werkstücken besser einschätzen. Menschen, die Roboter überwachen oder mit ihnen kooperieren, werden durch das IR-Licht nicht gestört.

Weißes Licht kann man nicht nur mit weißen LEDs erzeugen. Das geht genauso gut oder noch besser (aber teurer) mit RGB-LEDs. Hier sitzen LED-Chips mit (mindestens) den drei Primärfarben in einem Gehäuse. Durch unterschiedliche Ströme durch die diversen Chips kann man fast jede beliebige Farbmischung einstellen. In **Bild 9** ist beispielsweise eine OSTAR-SMD-LED von OSRAM zu sehen, die nicht nur je einen Chip mit roter (625 nm), grüner (530 nm) und blauer Farbe (453 nm) enthält, sondern auch noch eine weiße LED. Sie eignet sich für spezielle Beleuchtungszwecke wie etwa zur präzisen Beleuchtung bei medizinischen Eingriffen und da, wo es auf Farbverbindlichkeit ankommt. Trotz aller Kleinheit kommen bei 1,4 A Nennstrom über 500 Lm aus diesem SMD-Gehäuse.

Wer gelegentlich das Tanzbein schwingt, wird schon über das Vordringen von LEDs in Discos gestolpert sein. Sogenannte PAR-Kannen, also Bühnenscheinwerfer mit Farbfiltern, waren früher mit glühenden Lampen mit mehreren hundert Watt Leistung bestückt. Heute sind sie vielfach ledifiziert und voll elektronisch (**Bild 10**). Auch wenn LED-Versionen noch nicht ganz an die alte Technik heranreichen, wird der Unterschied durch immer hellere LEDs immer kleiner. Lebensdauer, Robustheit, weit niedrigere Hitzeentwicklung und geringere Verletzungsgefahren (platzende Scheinwerfer) sowie eine trägheitslose elektronische Steuerung sprechen eindeutig für die LED-Variante.

Doch farbiges LED-Licht wird nicht nur im Zusammenhang mit Musik geboten. Gerade Philips hat frühzeitig erkannt, dass auch Otto und Erna Normalverbraucher auf konfigurierbares Stimmungslicht stehen. Mit Lampen zum Auf-den-Tisch- oder In-die-Ecke-Stellen hat der niederländische Elektronik-Konzern vor einigen Jahren unter der Bezeichnung „LivingColors“ (**Bild 11**) eine ganze Serie von fernbedienbaren LED-Leuchten auf den Markt gebracht, deren Farbe sich sehr weitgehend einstellen lässt. Innendrin stecken etliche farbige und weiße LEDs, die nicht nur mischbar sind, sondern auch blinken und gedimmt werden können. Der Erfolg dieser Produkte hat Philips dann zu der Hue-Serie (**Bild 12**) motiviert. Unter Hue firmieren nicht nur Leuchten, sondern auch LED-Streifen und „normale“ farbige LED-Leuchtmittel mit E27-Sockel, die mit einer „Bridge“ genannten Fernsteuerung eingestellt werden. Unter „Hue Go“ werden Leuchten mit eingebautem Lithium-Akku verkauft. Es werden sicher noch weitere Varianten folgen, denn Philips verdient mit diesen Effektlichtsystemen richtig Geld. Doch auch wer weniger ausgeben will, wird fündig. Baumärkte haben das Geschäft gerochen und bieten ganze Regalwände und Grabbelkisten voller „Billigvarianten“ allerlei farbiger LED-Beleuchtung (**Bild 13**) für nur einen Bruchteil der Kosten von Markenprodukten an. Wer es noch billiger haben will, sucht in eBay oder bestellt via Alibaba direkt in China. Bei beiden Quellen muss

Bild 10. PAR-Effektscheinwerfer des Typs CLP56RGB05PS von Cameo mit 151 farbigen LEDs und einer Leistungsaufnahme von immerhin 30 W an 230 V. Der Scheinwerfer ist nicht nur mit einem DMX-Anschluss zur Fernsteuerung, sondern auch mit einem integrierten Mikrofon ausgestattet, was Lichtorgel-Effekte ermöglicht.

Bild 11. LivingColors von Philips: Ein Erfolgsmodell mit fernsteuerbaren farbigen und weißen LEDs in einer Kunststoffleuchte.

Bild 12. Zur Hue-Serie von Philips gehören farbige LED-Lampen mit E27-Fassung und die „Bridge“ als Fernsteuerung.

Bild 13. Jeder Baumarkt hat heute ein umfangreiches Sortiment an fernsteuerbaren und nichtfernsteuerbaren LED-Streifen, Leuchten und Effektlampen im Angebot.

Bild 14. Google Nexus 6p: Das OLED-Display dieses preiswerten Smartphones hat 2560x1440 Pixel.

man nicht nur mögliche Zoll-Komplikationen einkalkulieren, sondern auch in Kauf nehmen, dass billig nicht immer preiswert ist (siehe Kasten **Achtung Qualität!**). Selbstverständlich sind auch UV-LEDs zu etwas gut. Neben Spezialanwendungen kann man damit z.B. den fotoempfindlichen Lack beschichteter Platinen belichten oder aber „Schwarzlicht-Lampen" bauen. Eine andere Anwendung für ultraviolette LEDs ist die Beleuchtung eines mit Wasser durchflossenen transparenten Rohrs, um z.B. bei Aquarien und Swimming-Pools das Wachstum von Algen zu hemmen. Außerdem kann man damit Lacke aushärten oder die UV-Empfindlichkeit von Produktoberflächen testen.

OLEDs

OLEDs wurden zwar sehr viel später „erfunden" und können bezüglich Lichtausbeute und Langzeitstabilität noch nicht mit anorganischen Halbleiter-LEDs mithalten, doch sie holen zunehmend auf. Der Vorteile von OLEDs sind preiswertere Materialien und eine prinzipiell günstige Fertigungstechnik, da keine strengen Reinraumbedingungen erforderlich sind. Außerdem lassen sich OLEDs flächig fertigen, was für manche Beleuchtungszwecke von Vorteil ist, da so Schatten weich werden. Auch flexible Leuchtmittel sind kein prinzipielles Problem. Lange Zeit war es schwierig, konkrete OLED-Produkte jenseits des Labors herzustellen. Dies ändert sich aber zurzeit.

Mit OLEDs kann man z.B. sehr gut Displays fertigen, denn durch Drucktechnik lassen sich die organischen Leuchtpixel samt Verdrahtung kostengünstig herstellen, was leuchtstarke und kontrastreiche RGB-Displays auf OLED-Basis ergibt. Mit anorganischen Halbleitern wäre so ein Unterfangen sehr viel komplizierter und deutlich teurer. OLED-Displays konkurrieren zunehmend mit LCD-Displays. Prinzipiell schlägt ein OLED-Display seinen Flüssig-Kristall-Gegner in nahezu allen Punkten: Es ist keine Hintergrundbeleuchtung mit all den Ausleuchtproblemen notwendig, denn OLEDs leuchten von sich aus. Auch der Stromverbrauch ist geringer, da das Licht der Hintergrundbeleuchtung von LCDs durch die Lichtverteilung und die Polarisationsfilter zu großen Teilen geschluckt wird. Farbverfälschungen durch unterschiedliche Betrachtungswinkel gehören bei OLEDs der Vergangenheit an. Last not least ist der Kontrast von OLED-Displays uner-

Achtung Qualität!

LED-Leuchten, in denen für viel Licht viele einzelne LEDs eingebaut sind, gibt es schon eine ganze Weile. Schon seit Mitte der 1990er Jahre sind Autohersteller dazu übergegangen, auch bei Mittelklasse-Autos die sogenannte dritte Bremsleuchte mit einer Kette aus roten LEDs zu realisieren. Hierfür waren sie damals schon hell und preiswert genug. Der Kunde freut sich, dass er im Laufe eines Autolebens weniger Glühbirnchen wechseln muss. Außerdem ist eine Bremsleuchte sicherheitsrelevant und es ist daher gut, wenn sie nicht nach ein paar hundert Betriebsstunden ausfällt. Diese Überlegungen klingen schön und gut, aber das Szenario kriegt Risse, wenn es auf die Realität trifft.

Hier hat man es nämlich mit zwei Phänomenen zu tun, die zusammengenommen eine weniger ideale Wirklichkeit ergeben. Die in Bremsleuchten verbauten LEDs kommen „im Prinzip" auf enorme Lebensdauern von bis zu 50.000 Stunden. Doch im Auto klappt das garantiert nicht, denn hier gibt es nicht nur Vibrationen und gelegentlich schlechtes Wetter mit ordentlicher Luftfeuchtigkeit (Korrosionsgefahr für Leiterbahnen etc.), sondern auch extreme Temperaturschwankungen. Diese führen zu mechanischem Stress von LED-Gehäuse, Chip und Bonding. Ergo ist eine Lebensdauer von 50 kh Utopie. Die hohen Temperaturen des Blechs bei Sonnenschein sorgen zusammen mit der Betriebswärme für noch höhere Temperaturen. Folglich halten LEDs im Auto zwar immer noch länger als Glühbirnen, doch so groß ist der Unterschied nicht unbedingt. Verschärfend kommt noch der **Viele-LEDs-Effekt** hinzu: Selbst unter besten Bedingungen sinkt bei mehreren LEDs aus einfachen statistischen Gründen die Zeit, in der zu erwarten ist, dass noch **alle LEDs** funktionieren. Eine grobe Schätzung stellt die Reduktion der Gesamtlebensdauer um den Faktor $1/n^2$ dar. Und diese Korrektur macht schon bei 10 LEDs aus den 50 kh einer LED nur noch >15 kh für alle LEDs. Und sollte die reale Betriebsdauer im Auto auf 10 kh für eine LED sinken, muss man für die Zehn-LED-Kombi mit dem Ausfall einer LED nach etwa 3 kh rechnen. Das ist nun nicht nur deshalb schlimm, weil bei einer 12-V-Versorgung in der Regel etwa zwei bis drei LEDs in Reihe geschaltet werden und somit oft nicht nur eine LED dunkel wird. Schlimm ist, dass die LEDs in KFZ-Lampen hermetisch eingeschlossen und somit nicht austauschbar sind. Statt einem Birnchen für zwei Euro muss man dann schon einen größeren Schein in die Hand nehmen und die ganze Leuchte auswechseln. Dass dies keine akademischen Überlegungen sind, zeigt schon meine persönliche Erfahrung: Bei einem Fiat Baujahr 1996 war nach drei Jahren diese Bremsleuchte auf die beschriebene Weise defekt. Ersatzteilpreis plus Einbau lagen damals deutlich über 100 DM. Wer jetzt sagt: „Kein Wunder, ein Fiat!", dem sei erzählt, dass die vielgerühmte Zuverlässigkeit von Toyota ebenfalls exakt an einer LED scheiterte. Hier war es das linke Rücklicht meines Prius, das aus etlichen, fest eingebauten LEDs bestand. Nach 11 Jahren – zeitgleich mit dem ersten Auswechseln der Bremsscheiben – war also eine neue Rückleuchte fällig. Als preiswertes Nachbauteil waren hier zwar nur 73 € fällig, doch immerhin. Fallen bei neueren Autos LEDs in einem Frontscheinwerfer aus, so muss man mit Sicherheit mehrere hundert Euro auf den Tisch legen. Moderne Technik hat eben ihren Preis ;-).

Das Problem mit den vielen LEDs wird zum großen Problem, wenn man Leuchtmittel mit vielen LEDs im Haus hat. Hier spielen zwar Feuchtigkeit keine und Temperaturschwankungen eine untergeordnete Rolle. Dafür kommt es auf die Qualität der ver-

reicht, denn wenn eine LED aus ist, ist sie aus. Schwarz ist also wirklich schwarz. Bei all diesen tollen Eigenschaften wundert man sich, warum sich OLED-Displays noch nicht durchgesetzt haben. Der Grund liegt in den extremen Präzisionsanforderungen an die Fertigungstechnik: Die vielen LEDs eines Displays müssen nicht nur in engen Grenzen gleich hell sein, sondern das auch bleiben. Resultat ist, dass TV-Geräte auch mit kleinen Diagonalen vor zwei Jahren noch praktisch unbezahlbar waren. Doch Ende 2016 wurden vom OLED-Marktführer LG Geräte mit immerhin 55"-Diagonale halbwegs akzeptabel für unter 1.500 € angeboten. Brauchbare Smartphones mit OLED-Display gibt es aber schon länger und der Preisunterschied zur LCD-Variante fällt bei diesen kleinen Displays relativ gering aus, obwohl hier wie beim recht günstigen Smartphone Google Nexus 6p (**Bild 14**, gefertigt von Huawei) mit 2560x1440 Pixeln immerhin 11 Millionen OLEDs verbaut sind. Und wer sich die Frage stellt, warum es noch kein iPhone mit OLED-Display gibt: Dafür würden im Jahre 2016 schlicht und einfach die weltweiten Fertigungskapazitäten nicht ausreichen. Doch diese werden ständig ausgebaut. Laut Gerüchteküche soll es Ende 2017 soweit sein und das iPhone 8 wird mit einem OLED-Display bestückt – zumindest die besseren Modelle, denn für >100 Millionen Displays wird es auch dann wohl noch nicht reichen.

Die Zukunft

Neben dem iPhone 8 werden OLED-Displays in Kürze in so ziemlich allen Geräten zu finden sein, die etwas anzeigen wollen. Es ist bestimmt nicht zuviel behauptet, wenn man sagt, dass bald so ziemlich jede andere Display-Technik ausgedient hat. OLEDs werden auch in Beleuchtungskonzepte einfließen. Wenn sich das rechnet, wird nicht nur ein Leuchtenhersteller einsteigen. Leuchtmittel für konventionelle Fassungen aber bleiben die Domäne von anorganischen Halbleiter-LEDs, da bei OLEDs auf absehbare Zeit die Leuchtdichte nicht ausreicht, um auf kleinem Raum genügend Licht abzugeben. Anorganische LEDs werden bis auf Nischen mit speziellen Anforderungen so gut wie alle anderen Lichttechniken verdrängen. Dies nicht nur, weil sie energiesparender sind, sondern weil durch geringere Wartungskosten (seltenerer Leuchtmitteltausch) die Gesamtkosten deutlich niedriger als bei konventioneller Lichttechnik ausfallen. Straßenbeleuchtung ist da nur eine Anwendung, wo LEDs bald alternativlos sein werden. Mit LEDs lassen sich intelligente Fahrzeugscheinwerfer realisieren, die den Lichtkegel automatisch so modifizieren, dass entgegenkommende Fahrzeuge ausgespart und daher nicht geblendet werden. Dann ist es vielleicht nicht mehr notwendig, zwischen Abblend- und Fernlicht hin- und herschalten zu müssen. Schön wär's... ⏮

(160246)

Weblinks

[1] Artikel „Es werde LED!": www.elektormagazine.de/150577

[2] Stenkelfeld: www.youtube.com/watch?v=K_sIIUYLhZw

wendeten Bauteile an. Preiswerte Lichtketten aus Fernost für die Weihnachtsdeko oder „einfach so" sind eine super Sache. Bei etlichen Angeboten kann man den Preis kaum glauben: Die ganze Kette ist oft billiger als die Summe der Preise der verbauten LEDs. Die Frage: „Wie machen die das?" lässt sich leicht beantworten: Es wird nicht selten mindere Qualität verbaut. Nicht nur Lichtwettrüsten à la Stenkelfeld [2] kann Probleme machen – wer mit LED-Ketten zu tun hatte, hat auch Erfahrung mit (teilweise) ausgefallenen Lichterketten. Ich bin da keine Ausnahme :D.

Preiswerte LED-Streifen mit selbstklebender Rückseite, mit denen man so schön Möbel und Sonstiges indirekt mit Niederspannung beleuchten kann, haben das gleiche Problem. Den Vogel schossen bei mir zwei in Aluprofile eingeklebte LED-Streifen mit je 2 m Länge und insgesamt 120 LEDs ab. Nach nur sechs Monaten und vielleicht 600 Betriebsstunden waren schon acht LEDs ausgefallen! Nach der obigen Formel hätte ich den ersten Ausfall nach etwa 4.500 h erwartet. Nachdem ich von einem Reservestreifen einige gleichfarbige SMD-LEDs abgelötet und die defekten Exemplare damit ersetzt hatte, waren in zwei Monaten schon wieder zwei tot (**Bild 15**). A never ending Story...

Bild 15. LED-Streifen mit Ausfall. Am Tod von LED L30 bin ich nicht ganz unschuldig, denn so hochgewölbt wurde sie nicht gut gekühlt. Sie ist übrigens nicht hochohmig, sondern hat einen Kurzschuss. Ihre Vorgänger aber brauchten keinen Anlass zum spontanen Abgang.

Auch „normale" LED-Leuchtmittel halten oft weniger lange als sie sollten. So lange gibt es LED-Lampen ja noch nicht und trotzdem wurden bei mir schon vier davon mit E27-Gewinde und zwei Spots mit GU10-Sockel defekt. Bei den E27-Lampen starb die Vorschaltelektronik und bei den GU10-Spots die LEDs wegen zu hoher Temperaturen. Und bei zwei als Ersatz für Leuchtstofflampen gekauften LED-Röhren haben die Ingenieure von LG die Sicherung des integrierten Schaltnetzteils zu knapp dimensioniert. Nach Ersatz durch stärkere Sicherungen leuchten sie noch heute. Übrigens funktioniert auch meine vor über zehn Jahren selbst gebaute LED-Lampe (**Bild 16**) bis heute ohne Murren. Hier habe ich 13 LEDs guter Qualität verbaut.

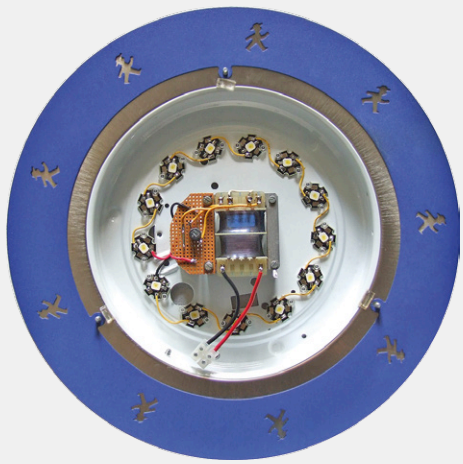


Bild 16. Die erste LED-Lampe im Hause Scherer aus dem Jahre 2005 war ein Selbstbau. Und sie hält bis heute – bestimmt schon über 10.000 Stunden.

ATtiny in Assembler

Kleine Mikrocontroller & Atmel Studio

Nicht für alles braucht man einen Arduino. Doch gerade kleine AVR-Controller haben wenig Speicher, weshalb eine besonders kompakte Programmierung hilfreich ist. Weder C noch BASCOM sind hier optimal. Die Programmierung in Assembler mit der IDE Atmel Studio liegt näher. In diesem Beitrag sehen Sie, wie das funktioniert.

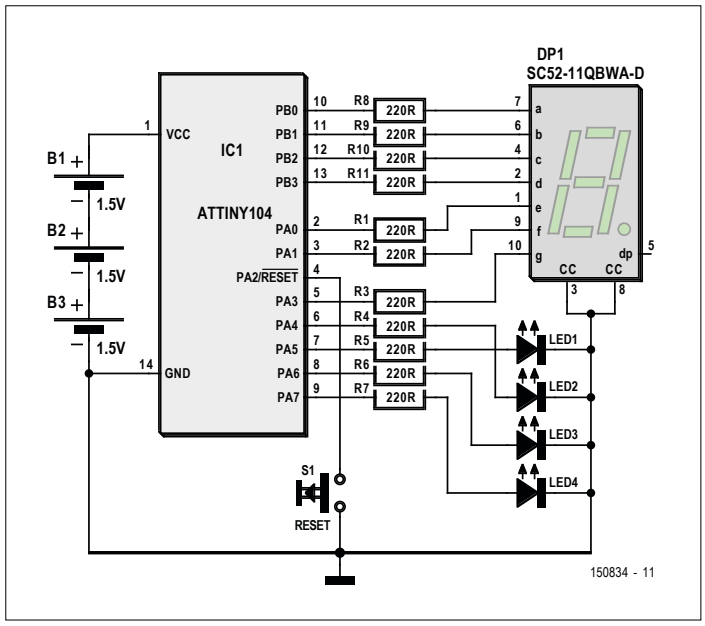


Bild 1. Die Schaltung des Zahnputz-Timers benötigt außer dem ATtiny104 nur wenige externe Bauelemente.

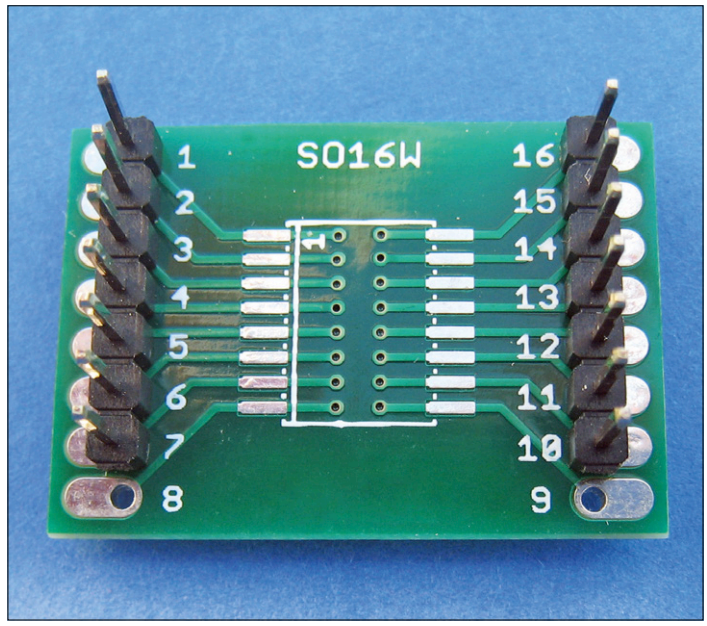


Bild 2. SMD-ICs passen nicht gut auf Lochraster-Platinen oder Steckbretter. Mit einer Adapterplatine auf ein DIL-kompatibles 1/10"-Raster klappt das viel besser. Hier werden nur 14 der 16 Pins verwendet.

Von **Miroslav Cina** (D)

Für manche Zwecke tun es einfache 8-bit-Mikrocontroller vollauf. Und es werden sogar heute noch immer wieder neue Varianten aufgelegt. Die beiden Typen ATtiny102 und ATtiny104 sind solche Neuerscheinungen, denn sie wurden erst Anfang 2016 vorgestellt. Im SOIC8- (102) bzw. SOIC14-Gehäuse (104) sind sie nicht nur klein, sondern haben auch nur 1 KB Flash-Speicher und sparsame 32 Byte RAM. Dafür glänzen sie mit anderen Fähigkeiten wie mehrkanaligem 10-bit-ADC, 16-bit-Timer und doppeltem PWM-Ausgang sowie seriellen Schnittstellen in Hardware (SPI und UART).
Gut geeignet sind solche Winzlinge für einfache Aufgaben bei geringem Stromverbrauch. Wem hier IoT einfällt, der liegt schon ganz richtig. Denn bei einer Rechenleistung von 1 MIPS/MHz ist eine Stromaufnahme ab 200 µA/MHz im Betrieb möglich. Im Idle-Modus sinkt der Strom auf Werte ab 25 µA/MHz und im Schlafmodus sind weniger als 0,2 µA drin. Mit aktivem Watchdog muss man mit rund 5 µA rechnen. Apropos Rechnen: Diese kleinen Controller sind je nach Stückzahl für unter 1 $ zu haben. Aus diesen Gründen sind die nachfolgenden Code-Beispiele auch für diesen Mikrocontroller maßgeschneidert. Das heißt aber nicht, dass man darauf beschränkt ist. Das hier vermittelte Know-How lässt sich auch prima bei anderen ATtiny- und sogar ATmega-Typen anwenden.

Beispielschaltung

Der bekannteste Beispiel-Code besteht sicherlich in der Ausgabe von „hello world" durch ein wie auch immer geartetes Computersystem. Da es hier zwar auch um Software, aber eben in einer hardwarenahen Variante mit einem Mikrocontroller geht, ist ein Mini-Projekt mit richtigen Bauteilen besser geeignet.
Als Hallo-Welt-Ersatz dient hier eine Timer-Schaltung, und zwar eine ganz Spezielle: Es geht um eine Zahnputzstoppuhr, welche die Kleinen (und die Großen) dazu anhalten soll, die Zahnpflegeaktivität ausreichend lange durchzuführen. Konkret: Die Stoppuhr soll mit einer Zeit von vier Minuten loslegen und rückwärts zählen. Dabei soll sie die aktuelle Zeit auf einem Siebensegment-LED-Display anzeigen. Da diese Zeitangabe recht grob ist, soll von insgesamt vier LEDs jeweils eine nach je 15 Sekunden verlöschen. Bei jeder vollen Minute sind wieder alle vier LEDs an. Leuchten also auf dem Display eine „1" und noch zwei LEDs, sollte man die Beißerchen noch 30 Sekunden bearbeiten. Der Countdown ist abgelaufen, wenn nichts mehr leuchtet. Für diese Aufgaben reicht eine Versorgung per Batterie.

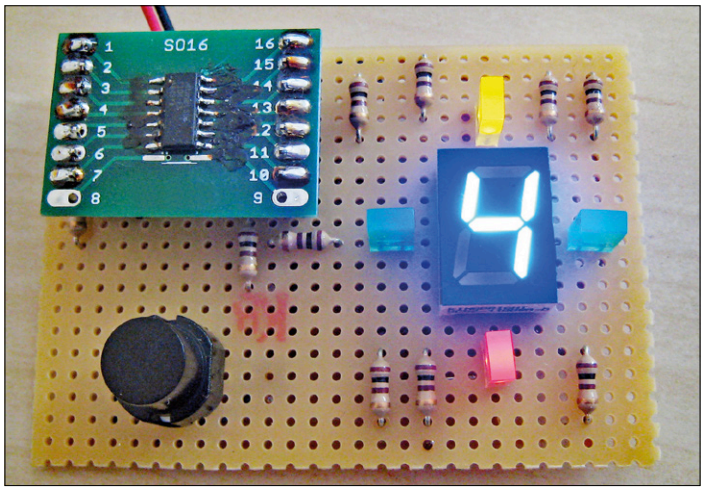


Bild 3. Der funktionierende Prototyp des Zahnputz-Timers.

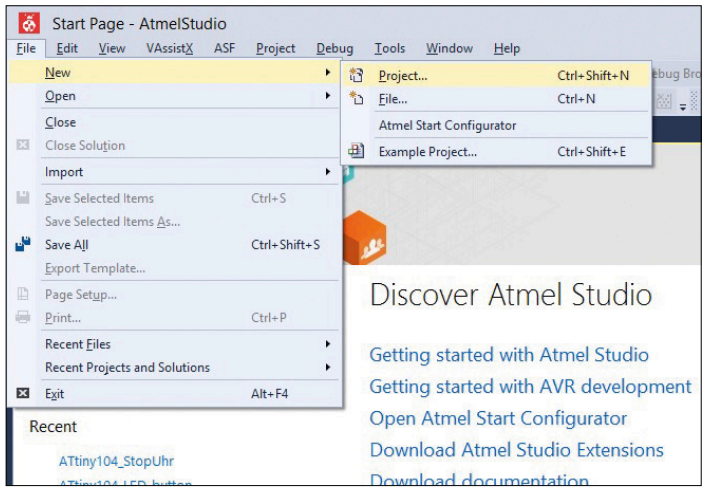


Bild 4. Anlegen eines neuen Projekts im Atmel Studio.

Wie man in der Schaltung von **Bild 1** sehen kann, ist außer dem Mikrocontroller nicht viel nötig: Hinzu kommen das Siebensegment-Display mit gemeinsamer Kathode, vier LEDs, elf Vorwiderstände und ein Taster. Da außer Masse und VCC noch zwölf Pins benötigt werden, braucht es die 14-polige Variante ATtiny104. Die komplette Elektronik wird durch drei Batterien (AA- oder AAA-Zellen) mit 4,5 V versorgt. Die Schaltung ist so simpel, dass man sie gut auf einem Stück Lochrasterplatine aufbauen kann. Da Experimentierplatinen mit ihrem 1/10"-Raster nicht so gut mit SMD-ICs bestückbar sind, eignet sich zur „Vergrößerung" des ATtiny104 eine handelsübliche Adapterplatine SOIC/DIL (siehe **Bild 2**). Der fertige Prototyp ist dann in **Bild 3** zu bewundern.
Die zwölf I/O-Pins des ATtiny104 sind in zwei Gruppen aufgeteilt: PA0...PA7 und PB0...PB3. Der Reset-Pin PA2 liegt am Taster S1. Wenn man ihn betätigt, wird der Mikrocontroller aus dem Schlaf geweckt, und es startet der Countdown. Die vier LEDs werden von PA4...PA7 angesteuert. Die restlichen sieben I/Os (PA0, PA1, PA3 und PB0...PB3) sind für die Steuerung von DP1 gedacht. Nach Ablauf des Countdowns sind die LEDs und das Display aus und der Mikrocontroller wird in den Power-Down-Modus versetzt. Anschließend liegt der Verbrauch bis zum nächsten Wecken mit S1 im Sub-µA-Bereich – also im Bereich der Selbstentladung der Batterien, weswegen diese dauerhaft angeschlossen bleiben.

Atmel Studio

Die Standard-IDE für AVR-Mikrocontroller ist Atmel Studio, aktuell in der Version 7. Diese IDE unterstützt nicht nur sehr viele Controller aus dem Hause Atmel, sondern diverse Programmiersprachen, darunter eben auch AVR-Assembler. Es lohnt sich schon wegen der Bugfixes, den neuesten Build (Ende 2016 Nr. 1188) herunterzuladen [2]. Außerdem ist dann auch sichergestellt, dass die neuesten Mikrocontroller unterstützt werden. Um diese IDE einzusetzen, sollte man nicht nur ein neues Projekt anlegen können, sondern auch wissen, wie man welche Befehle schreibt und den Code kommentiert.

Projekt anlegen

Nach erfolgreicher Installation startet man Atmel Studio 7.0 und legt via *File -> New -> Project* (Ctrl+Shift+N) ein neues Projekt an (siehe **Bild 4**). Anschließend wird festgelegt, dass es sich um ein Assembler-Projekt handelt und wo die zugehörigen Dateien gesichert werden (**Bild 5**). Nun muss noch der Mikrocontroller ATtiny104 ausgewählt werden (**Bild 6**). Nach Bestätigung mit „OK" wird das Projekt angelegt und dessen zentrales Element angezeigt: Der Assembler-Code in der Datei „main.asm" (**Bild 7**). Es wird ein kleines Beispiel mit einer Endlos-Schleife generiert, die immer wieder den Inhalt des

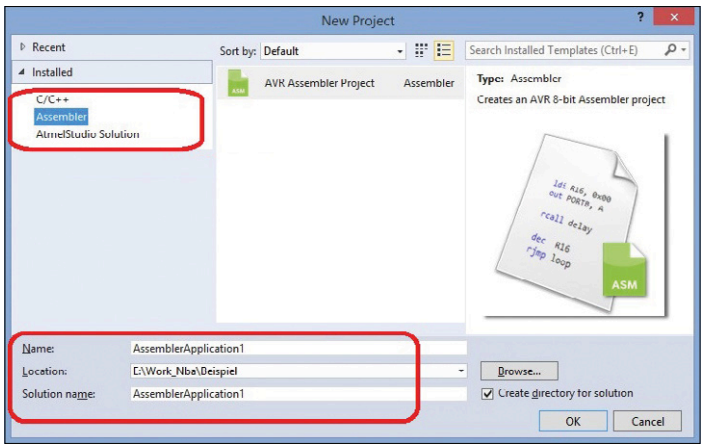


Bild 5. Festlegen von Projekt-Typ (Assembler) und Speicherort der Projektdateien.

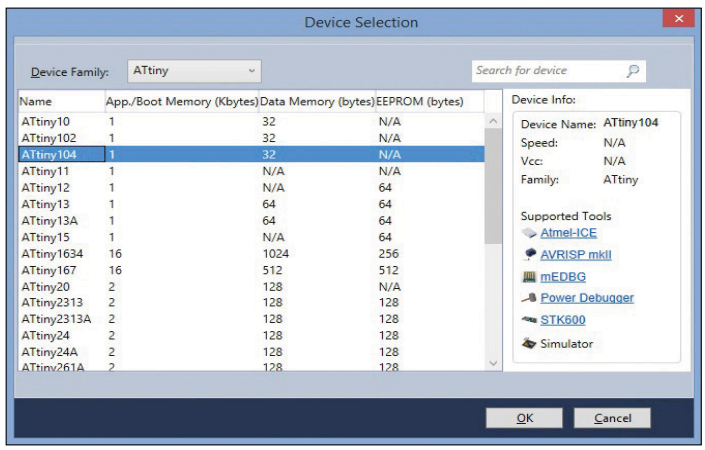


Bild 6. Auswahl des richtigen Mikrocontrollers.

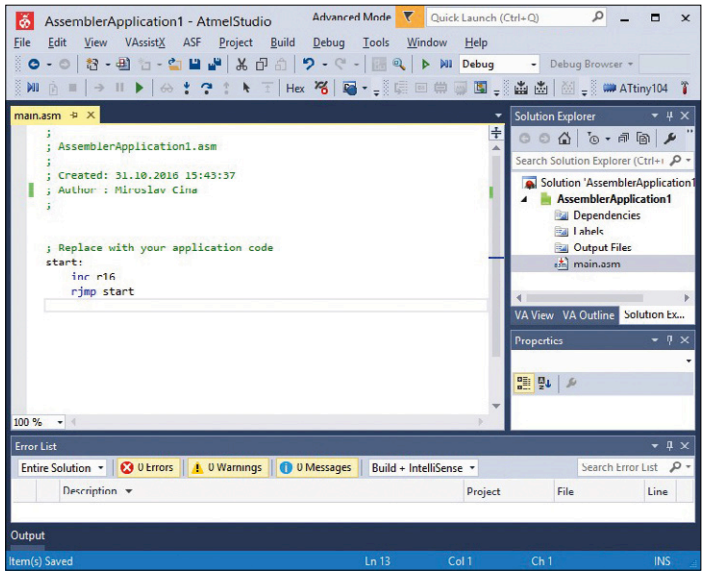


Bild 7. Automatisch erzeugter Beispiel-Code.

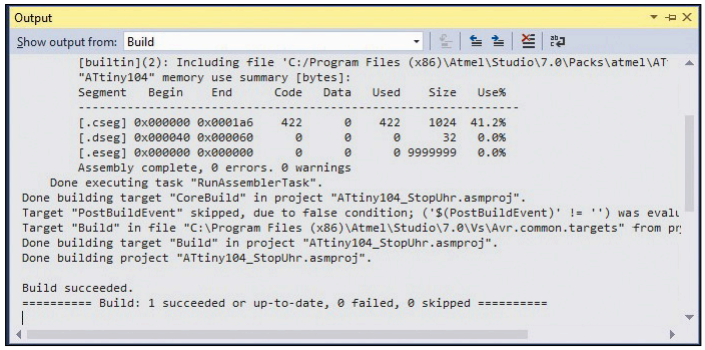


Bild 8. Ausgaben des Compilers.

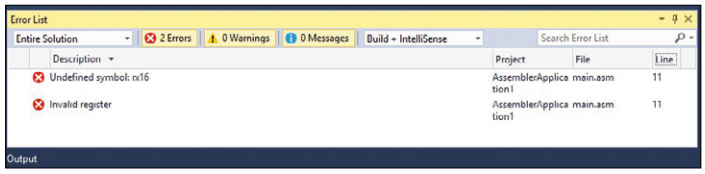


Bild 9. Fehlermeldungen.

Registers R16 inkrementiert. An diesem Beispiel kann man fast alles sehen, was man für „richtigen Code“ braucht: Vorhanden sind Kommentare in grün, mit „start:“ ein schwarzes Label und blaue Befehle samt schwarzen Argumenten.

Syntax

Wie man in Bild 7 sieht, kommt vor jeden **Kommentar** ein Semikolon. Der Text nach dem Semikolon bis zum Ende der Zeile wird als Kommentar aufgefasst, grün eingefärbt und somit ignoriert. Ein Kommentar kann auch nach einem Befehl, Label usw. stehen.

Labels sind ebenfalls recht einfach aufgebaut: Ein Name am Zeilenanfang, gefolgt von einem Doppelpunkt. Ihr Sinn liegt darin, dass man von einer anderen Codestelle per Sprungbefehl zum Label springen kann. In Bild 7 ist mit dem Sprung `rjump start` eine Endlos-Schleife realisiert.

Auch **Befehle** bereiten kein Kopfzerbrechen. Sie selbst werden blau dargestellt und das Argument schwarz: `inc r16`. Nun zur Schreibweise von **Zahlen**. Der folgende Befehl lädt die binäre Zahl 1110 1001 in das Register R16: `ldi r16,0b11101001`. In hexadezimaler Schreibweise wird die gleiche Wirkung mit dem Befehl `ldi r16,0xE9` erzielt. Bei binären Zahlen wird also ein „0b“ und bei hexadezimalen ein „0x“ vorangestellt. Für dezimale Zahlen braucht es kein Präfix.

Compiler-Lauf

Nach der Fertigstellung oder zwecks Fehlersuche kann man versuchen, den Code zu kompilieren. Hierfür wählt man im Menü „Build“ die Option „Build Solution“ aus oder betätigt F7. Als Ergebnis sollte man im Fenster „Output“ in etwa den in **Bild 8** dargestellten Inhalt sehen können. Falls noch irgendwo Bugs im Code stecken, wird es Fehlermeldungen wie in **Bild 9** hageln.

Assembler-Basics

Auch wenn man nur ein einfaches Assembler-Programm schreiben möchte, braucht es ein paar Grundinformationen über Assembler-Befehle und zudem über Mikrocontroller-Interna. Das Folgende wird aber kein umfassender Assembler-Kurs werden, es werden lediglich die erforderlichen Befehle abgehandelt. Für die Zahnputzstoppuhr verwenden wir nur die Befehle, die in **Tabelle 1** aufgelistet sind. Nachfolgend finden sich individuelle Erläuterungen ihrer in der Regel einfachen Syntax:

```
cbi  A, b
sbi  A, b
```

Dabei steht A für eines der 32 I/O-Register, weshalb A einen Wert von 0...31 haben darf. Das zu setzende oder zu löschende Bit wird durch den zweiten Parameter b spezifiziert. Da es sich um 8-bit-Register handelt, kann b den Wert 0...7 annehmen.

```
ldi  Rd, K
```

Der Index d spezifiziert eines von 16 Registern und kann einen Wert von 16...31 haben. Der Parameter K ist die Konstante, die in dieses Register geschrieben werden soll. Die 8-bit-Konstante darf Werte von 0...255 aufweisen.

```
out  A, Rr
```

Der Register-Parameter A kann Werte von 0...63 haben (es können Ports, Zähler, Konfigurationsregister etc. angesprochen werden). Der Parameter r indiziert das Ausgaberegister und kann Werte von 0...31 aufweisen.

```
rcall  k
```

Der Parameter k enthält die relative Adresse der Subroutine, zu der gesprungen werden soll. Beim kleinen Mikrocontroller ATtiny104 erreicht man damit problemlos den gesamten verfügbaren Speicherbereich.

```
ret
```

Der Befehl zum Rücksprung aus einer Subroutine zum nächsten Befehl des aufrufenden Codes benötigt keinen Parameter.

```
sleep
```

Auch hier ist kein Parameter vorgesehen. Es gibt verschiedene Befehle zum Aufwecken des Mikrocontrollers, doch nach einem Reset ist er ganz sicher nicht mehr im Schlafmodus.

`brne k`
Ist das Ergebnis der vorhergehenden Operation ungleich „0“, wird zu einer Stelle k relativ zum Programm-Counter gesprungen (PC + k). Der Parameter k kann Werte von -64...+63 aufweisen.

`dec Rd`
Bei der Dekrementierung spezifiziert der Index d das jeweilige Register, das den Wert 0...31 aufweisen kann.

ATtiny104-Basics

Für unser Projekt muss man die verwendeten Register des Mikrocontrollers kennen. Register sind Speicherzellen, die zum Rechnen und Zwischenspeichern von Werten benutzt werden können, andere Register steuern Hardware-Funktionen. Der ATtiny104 bietet insgesamt 16 Arbeits-Register, die 8 bit breit sind und mit R16...R31 bezeichnet werden. Von diesen 16 werden in diesem Projekt drei tatsächlich verwendet.
Die beiden I/O-Ports PA und PB des ATtiny werden in unserem Projekt beide benutzt. Man kann die Pins dieser Ports individuell als Ein- oder Ausgang konfigurieren, und zwar über die Register DDRA und DDRB (Data Direction Register von PA und PB). Die Werte der Bits von DDRx definieren, ob der zugehörige Pin als Ein- oder Ausgang konfiguriert ist. Ein Wert von „0“ macht den Pin zum Eingang und eine „1“ macht ihn zum Ausgang.
Den Status jedes Pins kann man individuell setzen oder löschen, hierfür sind die beiden Register PORTA und PORTB zuständig. Will man beispielsweise Bit 4 von PORTA auf „1“ setzen, schreibt man:

```
sbi  PORTA, 0x04
```

Firmware

Zunächst muss man überlegen, welche Funktionen die Firmware bereitstellen soll. Grundsätzlich benötigt auch Software für Mikrocontroller einen Initialisierungsblock. Eine Stoppuhr muss natürlich die Minuten auf dem Sieben-Segment-Display anzeigen können. Außerdem müssen hier noch vier LEDs im 15-Sekunden-Takt geschaltet werden. Zur eindringlicheren Wirkung wäre es nicht schlecht, wenn diese LEDs auch noch blinken würden. Jede Uhr muss mit Zeit umgehen können: Hier wird zumindest eine Routine benötigt, die eine Sekunde wartet. Zum Schluss muss man das Programm noch irgendwie beenden und den Mikrocontroller in den Schlaf schicken. Nachdem dies geklärt ist, kann man sich den einzelnen Funktionen widmen.

Initialisierung

Wichtig ist lediglich, dass die Portpins richtig als Ein- oder Ausgänge konfiguriert sind. Es werden ja alle I/O-Pins verwendet, und alle außer PA2 sollen Ausgänge sein. Nach einem Reset bzw. beim Einschalten sind alle Pins vorsichtshalber als Eingänge konfiguriert. In der Initialisierung muss also die Richtung aller Pins außer PA2 geändert werden. **Listing 1** zeigt, wie man das machen kann.
Beim Label „start“ geht es los. Im Block (1) sieht man, dass die vier LSBs von DDRB nacheinander auf „1“ gesetzt werden. Dadurch werden PB0...PB3 als Ausgänge konfiguriert.
In Block (2) kommen die Pins von PORTA an die Reihe. Das Zeile für Zeile abzuarbeiten wäre umständlich. Deshalb wird die Bit-Konfiguration für DDRA in Zeile (2.1) zunächst in Register R16 als Zwischenspeicher abgelegt. Alle Bits bis auf Bit 2 für PA2 erhalten den Wert „1“. In Zeile (2.2) wird dann der Wert von R16 auf einen Schlag in DDRA übertragen. Damit ist die Initialisierung auch schon fertig.

Listing 1.

```
;-----------------------------------------------
;Initialization
;-----------------------------------------------
start:
;(1) bits 0, 1, 2 and 3 of PORTB = output
   sbi   DDRB,0x00        ;set RB0 to output
   sbi   DDRB,0x01        ;set RB1 to output
   sbi   DDRB,0x02        ;set RB2 to output
   sbi   DDRB,0x03        ;set RB3 to output

;(2) all bits of PORTA except RA2 = output. RA2 = RESET
   ldi   R16,0b11111011   ;(2.1)
   out   DDRA,R16         ;(2.2)
;-----------------------------------------------
```

Listing 2.

```
;-----------------------------------------------
w_0:
   sbi   PORTB,0x00   ;Segment a
   sbi   PORTB,0x01   ;Segment b
   sbi   PORTB,0x02   ;Segment c
   sbi   PORTB,0x03   ;Segment d
   sbi   PORTA,0x00   ;Segment e
   sbi   PORTA,0x01   ;Segment f
   cbi   PORTA,0x03   ;Segment g
ret
;-----------------------------------------------
```

Listing 3.

```
;-----------------------------------------------
w_30:
   sbi   PORTA,0x07   ;LED4
   sbi   PORTA,0x06   ;LED3
   cbi   PORTA,0x05   ;LED1
   cbi   PORTA,0x04   ;LED2
ret
;-----------------------------------------------
```

Listing 4.

```
;-----------------------------------------------
delay1:
   ldi   R17,255
d_loop2:
   ldi   R16,164
d_loop1:
   dec   R16
   brne  d_loop1   ;inner loop
   dec   r17
   brne  d_loop2   ;outer loop
ret
;-----------------------------------------------
```

Zeichensatz

Um Zahlen auf einem Sieben-Segment-Display darzustellen, benötigt man so etwas wie einen Zeichensatz, wenn man die Bits für die einzelnen Segmente nicht jedes Mal händisch setzen will. Da die Stoppuhr nur Zahlen von 0...4 kennt, muss der Zeichensatz auch nur die Werte für diese vier Ziffern enthalten. Zur Darstellung der aktuellen Zeit ist es sinnvoll, ein paar Subroutinen zu schreiben, die man immer zur Zeichendarstellung verwenden kann.

In der Firmware gibt es folglich fünf passende Subroutinen „w_0“ bis „w_4“ (write 0 bis write 4). Aus der Schaltung kann man entnehmen, dass es sich um ein Display mit gemeinsamer Kathode handelt, dessen Segmente mit jeweils einem Mikrocontroller-Pin verbunden sind. Daraus ergibt sich, dass die Pins bei einem Wert von „1“ ihr jeweiliges Segment leuchten lassen, während es bei einer „0“ dunkel bleibt.

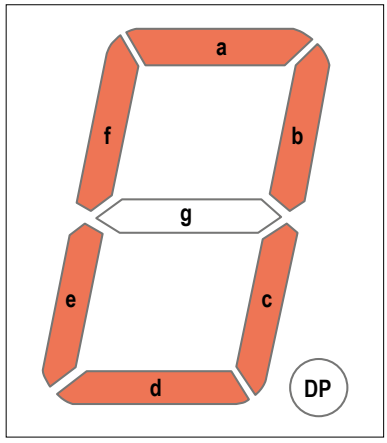


Bild 10. Beleuchtete Segmente des Displays zur Anzeige einer „0“.

In **Bild 10** sieht man, dass bei einer darzustellenden „0“ alle Segmente außer „g“ und dem nicht angeschlossenen Dezimalpunkt leuchten. **Listing 2** zeigt, wie die zugehörige Subroutine „w_0“ aussehen kann. Die Pins für die Segmente a, b, c, d, e und f werden per Befehl „sbi“ auf „1“ gesetzt und leuchten somit. Segment g wird sicherheitshalber gelöscht, falls es vorher aktiv war. Wie man auf die gleiche Weise die Subroutinen „w_1“ bis „w_4“ schreibt, ist logisch und muss nicht weiter erläutert werden. Der Zeichensatz ist also implizit in diesen fünf Routinen enthalten.

15-Sekunden-LEDs

Mit diesem Vorwissen ist die Ansteuerung der vier LEDs ein Kinderspiel. Auch hier gibt es fünf Subroutinen mit den Bezeichnungen „w_60“, „w_45“, „w_30“, „w_15“ und „w_00“. Sie schalten vier, drei, zwei, eine oder keine LED an. **Listing 3** enthält als Beispiel die Subroutine „w_30“, die zwei LEDs ein- und zwei LEDs ausschaltet. Sie ähnelt weitgehend den Subroutinen, welche die LED-Segmente des Displays ansteuern.

Listing 5.

```
;-------------------------
one_minute:
   ldi     R18,0x07   ;(1)
x_1:
   rcall   w_60       ;(2)
   rcall   wait_1s    ;(3)
   rcall   w_45       ;(4)
   rcall   wait_1s    ;(5)
   dec     R18        ;(6)
   brne    x_1        ;(7)
   rcall   w_60       ;(8)
   rcall   wait_1s    ;(9)
;-------------------------
```

Damit die LEDs zur richtigen Zeit ein- und ausgeschaltet werden bzw. blinken können, braucht es eine Zeitbasis. Als solche fungiert Subroutine „delay1“ von **Listing 4**, die einfach nach einem Aufruf etwa 125 ms wartet, bis sie wieder zurückspringt. Ruft man sie acht Mal auf, ist eine Sekunde vorbei. Ihre Besonderheit ist, dass sie eine Schleife in einer Schleife enthält, um viel Zeit zu schinden. Die innere Schleife wird 164 Mal durchlaufen (Zähler ist R16) und die äußere Schleife glatt 255 Mal (Zähler ist R17).

Eine Minute bitte

Das Display soll ja beim Countdown jede Minute die angezeigte Zeit ändern. Außerdem ändert sich der LED-Status alle 15 Sekunden (es verlöscht jeweils eine LED und eine andere fängt an zu blinken). All dies steckt in der Subroutine „one_minute“ von **Listing 5**.

Es werden sieben Mal ein paar Befehle (eigentlich nur Subroutinenaufrufe) wiederholt. Das passiert in der Schleife, die mit Label „x_1“ beginnt (2) und bis zum bedingten Sprung (7) geht. Am Anfang (1) wird die Konstante „7“ in R18 geladen. Dieser Wert gibt an, wie oft die Schleife durchlaufen wird, also werden sieben Mal bestimmte andere Subroutinen aufgerufen. Bei (2) wird „w_60“ angesprungen (alle vier LEDs einschalten), danach wird gleich eine Sekunde gewartet (3). Anschließend wird „w_45“ ausgeführt, somit die höchstwertige LED ausgeschaltet und dann wieder eine Sekunde gewartet (5). In Schritt (6) wird der Inhalt von R18 dekrementiert und falls die Null noch nicht erreicht ist, erfolgt ein Sprung (7) zurück auf „x_1“ (2). Jetzt sind wieder alle vier LEDs eingeschaltet. Es wird ständig zwischen „w_60“ und „w_45“ gewechselt, was das Blinken der höchstwertigen LED zur Folge hat. Nach sieben Durchläufen sind 14 Sekunden vorbei. Jetzt werden wieder alle LEDs aktiviert (8) und noch eine Sekunde gewartet (9), so dass die Sequenz insgesamt 15 Sekunden dauert.

Wenn dies vorbei ist, geht es weiter mit einer

Listing 6.

```
;-------------------------------------------------
;Countdown
;-------------------------------------------------
   rcall   w_4          ;(1)
   rcall   one_minute
   rcall   w_3          ;(2)
   rcall   one_minute
   rcall   w_2          ;(3)
   rcall   one_minute
   rcall   w_1          ;(4)
   rcall   one_minute
;-------------------------------------------------
```

```
;Done - just a effect to finalize the run
   rcall   w_0          ;(5)
   rcall   wait_1s
   rcall   fin1
;
   rcall   fin1
   rcall   fin1
;switch everything off and goto sleep
   rcall   w_space      ;(6)
   rcall   w_00         ;(7)
   sleep                ;(8)
;-------------------------------------------------
```

ähnlichen Routine hinter dem (nicht abgedruckten) Label „x_2“. Jetzt blinkt für weitere 15 Sekunden die nächsthöchstwertige LED (es wird zwischen „w_45“ und „w_30“ gewechselt). Anschließend kommt Label „x_3“ und „x_4“, bis die ganze Minute durch ist.

Hauptroutine

Die eigentliche Ablaufsteuerung der Stoppuhr steckt in der Hauptroutine von **Listing 6**. Beim Beginn des Countdowns wird die „4“ angezeigt (1) und dann läuft die Routine „one_minute“ durch. Anschließend wird die „3“ angezeigt (2) und wieder „one_minute“ aufgerufen und so weiter.
Wenn die vier Minuten abgelaufen sind, wird nur noch die „0“ angezeigt (5) und noch ein paar Mal die Subroutine „fin1“ aufgerufen, die das Display zum Zeichen des Ablaufs der Zahnputzzeit blinken lässt. Zum Abschluss werden alle Segmente des Displays gelöscht (6) und alle LEDs deaktiviert (7) sowie der Mikrocontroller in den Schlaf geschickt (8). Mehr passiert nicht. Erst nach einem Reset startet der Countdown neu.

Programmierung

Nun bleibt nur noch die Aufgabe, das fertige und kompilierte Programm in den ATtiny104 zu übertragen. Den kompletten Projekt-Code kann man wie immer kostenlos von der Elektor-Webseite zu diesem Artikel [3] downloaden. Der ATtiny104 unterstützt das sogenannte TPI (Tiny Programming Interface). Dieses verwendet außer VCC und GND lediglich noch die drei Leitungen TPI CLK, TPI Data und RESET. Es gibt einige Programmiergeräte, die dieses Interface unterstützen. Gut geeignet ist das „Xplained Nano Evaluation Kit“ [4] (Rückseite in **Bild 11**) – ein Evaluationsboard mit ATmega32U4 und ATtiny104 an Bord. Der erste Controller dient als USB-Programmer für den zweiten Chip, mit Hilfe des Atmel Studio. Selbstverständlich kann man damit auch einen angeschlossenen, externen ATtiny programmieren.
Dieses Board besitzt außerdem noch einen Taster (an PB1 angeschlossen) und eine LED (an PA5 angeschlossen). Mit Taster und LED kann man prima einfache Experimente durchführen, ohne dass man externe Hardware anschließen muss.
Wenn das Xplained-Board zum ersten Mal an einen PC angeschlossen wird, erkennt es der PC als „mEDBG CMSIS-DAP“. Außerdem sollte ein „Device Setup“ erscheinen. Atmel Studio erkennt das Board automatisch. Anschließend zeigt Atmel Studio die Board-Bezeichnung samt Seriennummer an. Um den ATtiny der Stoppuhr zu flashen, muss dieser über die schon erwähnten fünf TPI-Leitungen angeschlossen werden. Steckt der ATtiny schon auf einer Adapterplatine, ist eine Verbindung der ersten vier Pins dieser Platine mit VCC, PA0, PA1 und PA2 (in dieser Reihenfolge) des Xplained Nano kein Problem (Bild 11). Jetzt muss nur noch Pin 14 des ATtiny mit GND der Xplained-Platine verbunden werden. Wenn dann im Atmel Studio *Debug -> Start Without Debugging* gewählt wird, überträgt das Xplained-Board die kompilierte Firmware in den ATtiny104 – und zwar in beide, also auch in den ATtiny auf der Xplained-Platine. Das macht aber nichts. Hauptsache der Mikrocontroller der Stoppuhr ist fertig und kann in die Zielschaltung eingesetzt werden.

Fazit

Ich war überrascht, wie einfach man kleine Mikrocontroller von Atmel mit Assembler programmieren kann. Da ich keinen

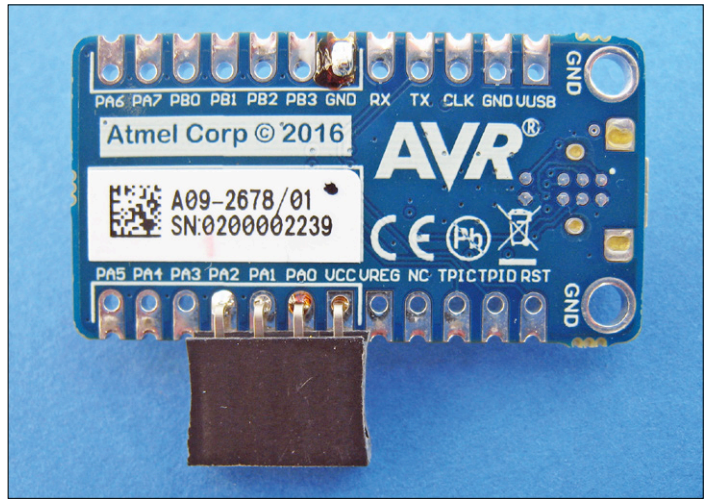


Bild 11. Rückseite des Xplained-Nano-Boards mit aufgelöteter vierpoliger Buchsenleiste; zum Anschluss eines ATtiny104 auf einer Adapterplatine.

speziellen Programmer besaß, habe ich einfach das Xplained-Board für diesen Zweck verwendet, was einfacher war, als ich zunächst dachte. Dieses Evaluation-Board ist ja schon deshalb interessant, weil man es bei den entsprechenden Distributoren für preiswerte 10 € bekommen kann.
Mögliche Rückfragen zum Thema beantworte ich gerne per Mail an miroslav.cina@t-online.de. ⏮

(150834)

Weblinks

[1] ATtiny102/104: www.atmel.com/devices/ATTINY104.aspx
[2] Atmel Studio 7: www.atmel.com/tools/atmelstudio.aspx
[3] Firmware: www.elektormagazine.de/150834
[4] Xplained Nano: www.atmel.com/tools/ATTINY104-XNANO.aspx

Tabelle 1.

Befehl	Beschreibung	Bedeutung
cbi	Clear Bit in I/O-Register	1 Bit von Port PA oder PB wird auf „0“ gesetzt
sbi	Set Bit in I/O-Register	1 Bit von Port PA oder PB wird auf „1“ gesetzt
ldi	Load Immediate	Konstante wird in ein Register geladen
out	Out to I/O-Port	Inhalt eines Registers an PA oder PB ausgeben
rcall	Relative Call to Subroutine	Relativer Sprung zu einer Subroutine
ret	Return from Subroutine	Rücksprung aus einer Subroutine
sleep	Sleep Mode	Versetzt Mikrocontroller in Schlafmodus
brne	Branch if Not Equal	Bedingter Sprung wenn ungleich
dec	Decrement	Inhalt eines Registers wird um „1“ verringert

Digitale Filter einfach erklärt

CIC-Filter

Von **Robert Lacoste** (Frankrijk)

In der letzten Folge haben wir die digitalen Filter mit endlicher Impulsantwort besprochen: *Finite Impulse Response*, FIR. Mit FIR-Filtern lassen sich beliebige Frequenzcharakteristiken realisieren, an die Signalprozessoren werden jedoch hohe Anforderungen gestellt. Diesmal geht es um die CIC-Filter, wobei CIC für *Cascaded Integrator Comb* steht. Diese Filter kommen mit weniger leistungsstarken Signalprozessoren aus, sie sind jedoch auch weniger flexibel. Häufig werden CIC-Filter eingesetzt, um die nötige Prozessorleistung zu reduzieren. CIC-Filter sind fortlaufend damit beschäftigt, gleitende Mittelwerte zu berechnen, und zwar auf transparentem und effizientem Weg. Diesmal geht es auch noch um einen wichtigen Begriff aus der digitalen Signalverarbeitung: *Multirate*.

Ein typisches Beispiel

Das Beispiel, mit dem wir beginnen wollen, ist ein Mono-Audiosignal in HiFi-Qualität, das mit 44,1 kS/s abgetastet wird (44100 Samples in der Sekunde, Audio-CD). Nach dem Theorem von Nyquist ist diese Abtastfrequenz geeignet, das Frequenzspektrum von Null (Gleichspannung) bis etwas weniger als 44,1 kHz / 2 = 22,05 kHz abzubilden. Zusammen mit einem steilen Antialias-Filter ist die Konfiguration besonders geeignet für Anwendungen im Audiobereich. Das Signal wollen wir digital auswerten, um Signalanteile mit Frequenzen unter 100 Hz zu erkennen, beispielsweise um elektronische Musik zu produzieren oder langsame Schwingungen aufzuspüren. Wir müssen die Signalanteile von Null bis 100 Hz ausfiltern, das sind 100 Hz / 44100 Hz = 0,2 % des betrachteten Frequenzbands. Bewerkstelligen lässt sich dies mit einem FIR-Filter, doch ein so schmales Filter erfordert viele „Taps“ (siehe vorausgegangener Beitrag [1]). Angenommen, es sind 256: Das FIR-Filter muss dann für jeden Abtastwert 256 Multiplikationen und Additionen durchführen, und zwar 44100 Mal in der Sekunde. Insgesamt sind das etwas mehr als 22 Millionen mathematische Operationen in der Sekunde, und das ist bestimmt nicht wenig!

Multirate-Verfahren

Wie im vorstehenden Beispiel enthalten auch reale Signale Unmengen an Informationen, von denen nur ein kleiner Teil von Bedeutung ist. Ziel ist es nun, das Sig-

nal möglichst schnell auf den relevanten Teil zu reduzieren, damit dieser Teil möglichst effizient verarbeitet werden kann. Im vorstehenden Fall wird die Abtastrate 44,1 kS/s vom Frequenzumfang des Audiosignals bestimmt (0...20 kHz), doch der Bereich der relevanten Information ist nur 100 Hz breit. Da hier 99,8 % der Informationen nicht von Interesse sind, treffen ebenso viele mathematische Operationen ins Leere. Durch Anwenden von Multirate-Algorithmen lässt sich diese Situation deutlich verbessern. Wie wir gesehen haben, wird das originale Signal mit hoher Rate abgetastet, doch nur ein kleiner Teil enthält die für uns nützlichen Informationen. Wenn wir nun schrittweise die Abtastrate verringern, bleiben die Nutzinformationen erhalten, während die Informationsdichte steigt (**Bild 1**). Bei hoher Abtastrate müssen die angewendeten digitalen Algorithmen einfach strukturiert sein, damit der Signalprozessor nicht an seine Grenzen stößt. Ist jedoch die Abtastrate herabgesetzt, sind auch komplexe Algorithmen anwendbar. Beim Generieren von Signalen können die gleichen Schritte in umgekehrter Folge durchlaufen werden, um die Abtastrate eines Signals zu erhöhen und die Informationsdichte herabzusetzen. Die Software des Signalprozessors oder FPGAs muss zwar in Echtzeit mit Datenströmen unterschiedlicher Geschwindigkeiten umgehen, doch dieses Problem lässt sich, wie wir sehen werden, mit einem Multirate-Verfahren lösen.

Dezimierung

Kurz zusammengefasst: Von einem mit 44,1 kS/s abgetasteten Signal, das ein 20 kHz breites Audiospektrum enthält, ist für uns nur der Frequenzbereich 0...100 Hz relevant. Deshalb wollen wir die Abtastrate auf etwas mehr als 2 · 100 Hz = 200 S/s verringern. In der digitalen Signalverarbeitung wird das Herabsetzen der Abtastrate als *Dezimieren* bezeichnet. Wie lässt sich die Abtastrate von 44100 S/s auf beispielsweise 220 S/s reduzieren? Mathematisch ist 44100 / 220 ≈ 200, es genügt folglich, einen von 200 Abtastwerten zu behalten und die übrigen 199 Werte zu eliminieren! Die Methode führt tatsächlich zum Ziel, allerdings unter einer Voraussetzung: Vor der Dezimierung müssen wir uns vergewissern, dass das Signal mit der Abtastrate 220 S/s zuverlässig reproduzierbar ist. Das bedeutet, dass darin keine Signalanteile vorkommen dürfen, deren Frequenzen höher als 220 Hz / 2 = 110 Hz sind. Wenn diese Voraussetzung nicht erfüllt ist, tritt im Signal nach der Dezimierung mehr oder weniger starkes Aliasing auf. Deshalb muss das Signal vor der Dezimierung einen Tiefpass mit der Eckfrequenz 100 Hz durchlaufen. Damit kehren wir jedoch zum ursprünglichen Problem zurück: Für das mit 44100 S/s abgetastete Signal ist ein Tiefpass nötig, bevor die Abtastrate durch Dezimieren reduziert werden kann! Dafür brauchen wir einen Tiefpass-Algorithmus, der dem Signalprozessor nur wenig Leistung abverlangt...

Gleitende Mittelwerte

Sicher ist Ihnen bekannt, dass Rauschen durch Bilden von Mittelwerten aus digitalen Signalen entfernt werden kann, da schnelle Änderungen geglättet werden und langsame Änderungen erhalten bleiben. Mit anderen Worten: Das Bilden von Mittelwerten, oder besser von *gleitenden* Mittelwerten, ist eine unspektakuläre Methode, einen Tiefpass zu realisieren. Der Algorithmus zum Bilden gleitender Mittelwerte ist denkbar unkompliziert: Aus N aufeinander folgenden Abtastwerten wird der Mittelwert berechnet, er ist bis auf den Faktor N gleich der Summe aus den N Abtastwerten. Die Summe

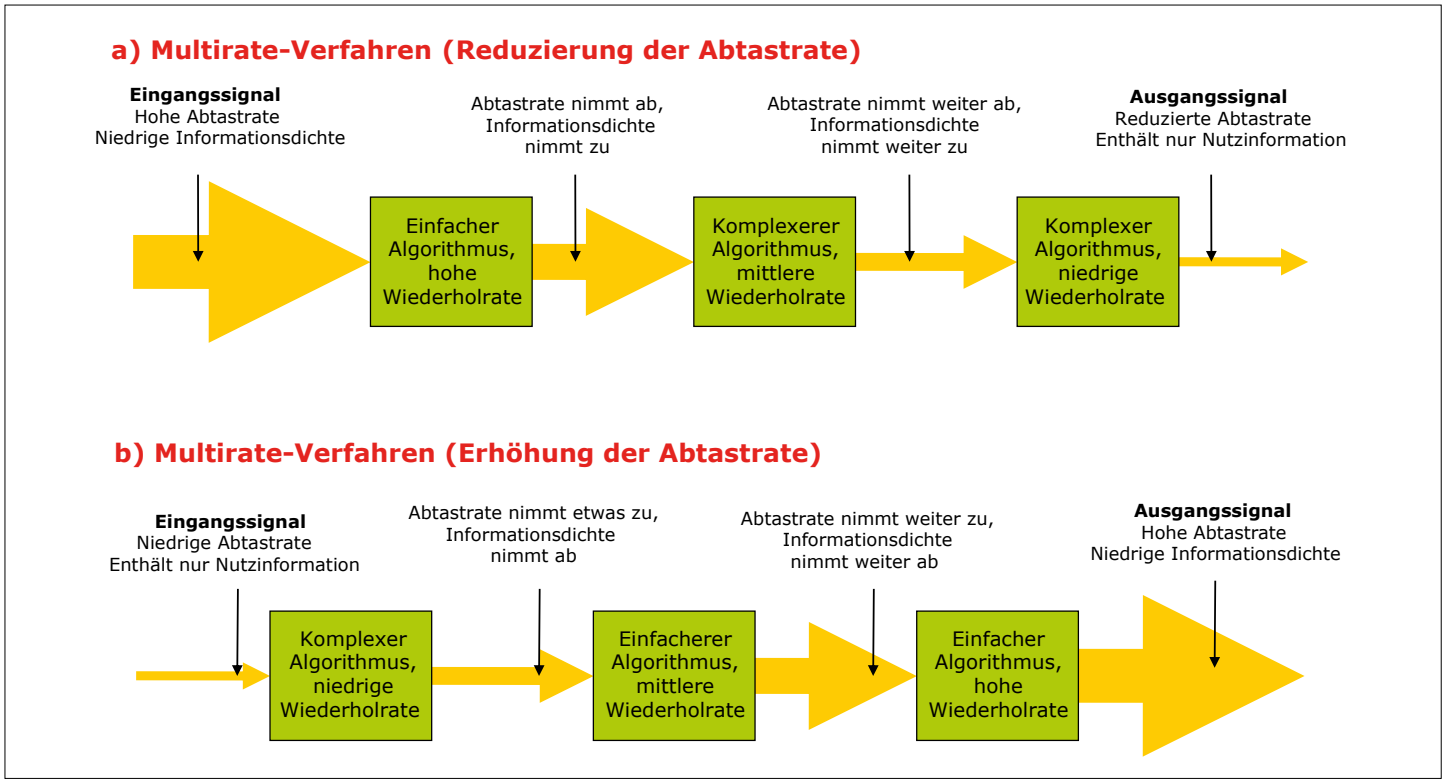


Bild 1. a) Die Multirate-Signalverarbeitung ist eine Methode, bei der die Abtastrate eines Signals stufenweise herabgesetzt wird, während die Dichte der relevanten Informationen steigt. b) Umgekehrt kann der gleiche Weg beschritten werden, um die Abtastrate eines Signals zu erhöhen.

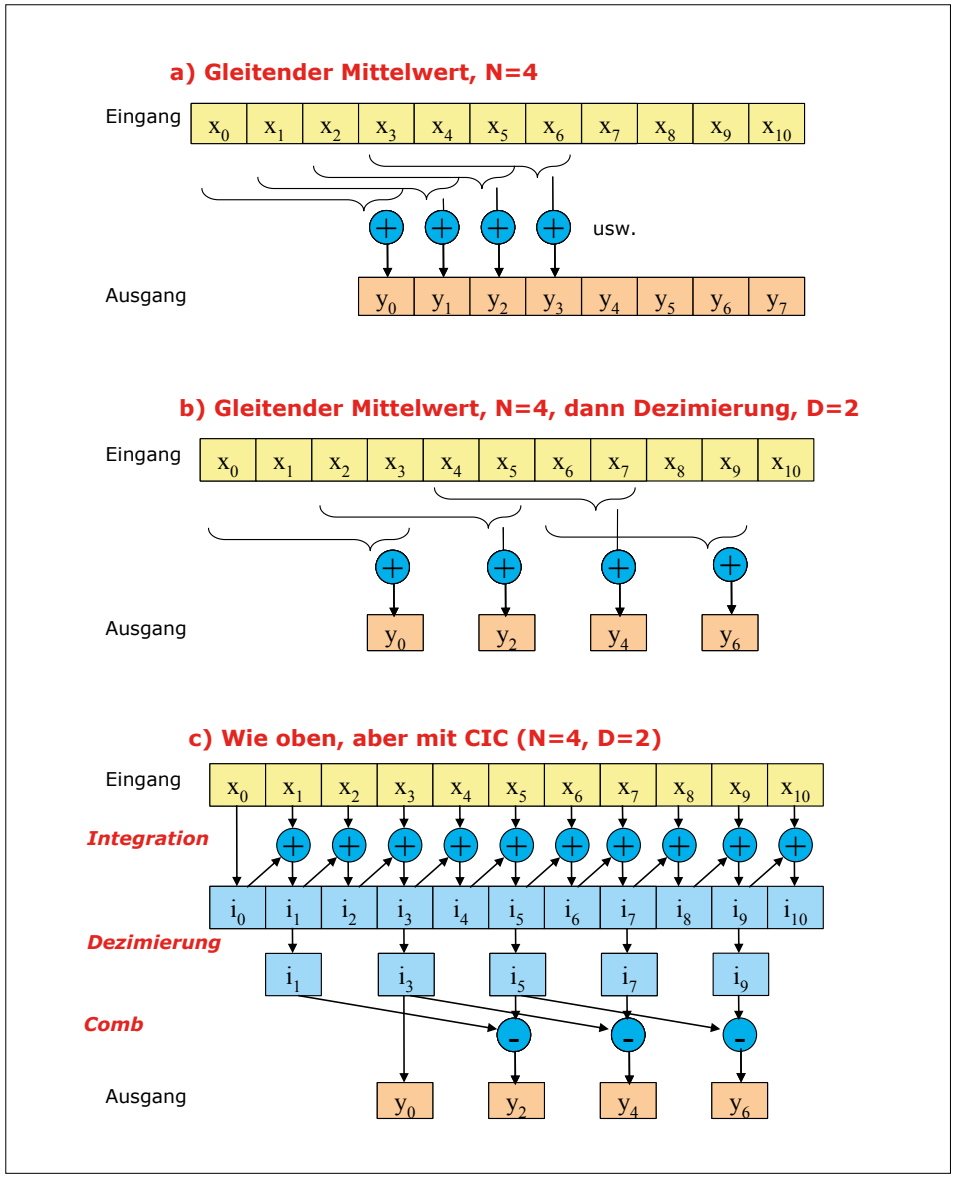


Bild 2. a) Bilden eines gleitenden Mittelwerts, hier mit N = 4 aufeinander folgenden Abtastwerten.
b) Das Dezimieren, hier mit D = 2, läuft auf das Löschen jedes zweiten Abtastwerts hinaus.
c) Ein CIC-Filter liefert exakt das gleiche Ergebnis, es arbeitet jedoch deutlich effizienter.

liefert folglich das gefilterte Signal. Zur Demonstration ist in **Bild 2a** ein Beispiel mit N = 4 dargestellt.

Da zum Bilden gleitender Mittelwerte nur Additionen erforderlich sind, können damit unkompliziert Signalanteile hoher Frequenzen aus Signalen gefiltert werden. Bei geschickter Wahl von N bleiben Signalanteile unter 100 Hz nahezu unverändert, während Signalanteile über 100 Hz stark abgeschwächt werden. Der Wert N = 256 ist hier ein guter Startwert. Anschließend können wir die Abtastrate mit einem Dezimationsschritt von beispielsweise D = 64 auf 689 S/s reduzieren. Es dürfte deutlich sein, dass es nicht sinnvoll ist, alle gleitenden Mittelwerte zu berechnen, bevor die Dezimierung stattgefunden hat. Nur jene Mittelwerte müssen berechnet werden, die bei der Dezimierung erhalten bleiben. In unserem Beispiel ist die Summe von N = 256 Abtastwerten zu berechnen, und dies 44100 / 64 = 689 mal in der Sekunde, was 256 · 689 = 176384 Additionen in der Sekunde ergibt. Diese Anzahl liegt erheblich niedriger als die zuvor veranschlagten 22 Millionen mathematische Operationen!

In **Bild 2b** ist das Bilden gleitender Mittelwerte über vier Abtastwerte (N = 4), gefolgt von einer Dezimierung um den Faktor D = 2, schematisch skizziert. Dort wird über zwei Abtastwerte hinweg der Mittelwert berechnet. Bei genauer Betrachtung fällt allerdings auf, dass der Algorithmus wenig effizient ist, weil wiederholt identische Berechnungen durchgeführt werden. Zum Beispiel ist der erste Ausgangswert $x_0+x_1+x_2+x_3$, während der zweite Ausgangswert $x_2+x_3+x_4+x_5$ beträgt. In beiden Ausgangswerten kommt die Summe $\mathbf{x_2+x_3}$ vor. Wenn der Mittelwert über mehr Abtastwerte berechnet wird, was in der Praxis häufig zutrifft, tritt dieser Effekt noch stärker in Erscheinung. Berechnen wir gleitende Mittelwerte über N = 256 Abtastwerte mit der Dezimierung D = 64, werden bestimmte Additionen 16 Mal wiederholt!

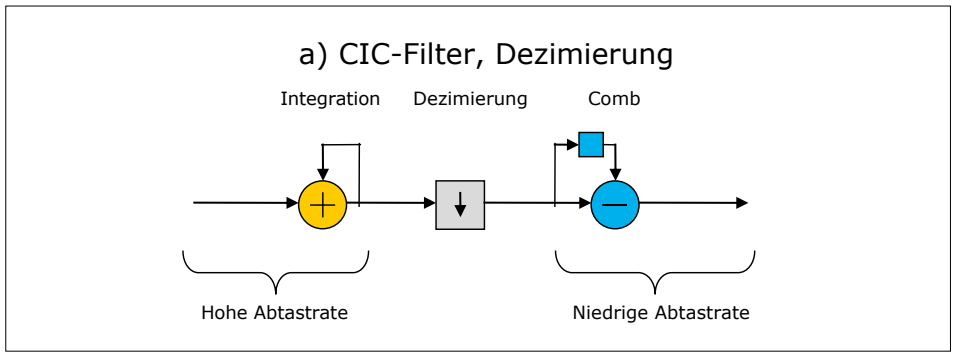


Bild 3. Zu einem CIC-Filter erster Ordnung (CIC1) gehören drei Stufen: Integrator, Dezimator und Comb.

Endlich: Das CIC-Filter!

Wir können die vorstehenden Berechnungen optimieren, indem wir festlegen, dass die Länge N des Mittelwerts ein Vielfaches des Dezimierungsfaktors D sein soll. Diese Überlegung führt auf direktem Weg zum CIC-Filter. In **Bild 2c** ist dieses Filter in seiner simpelsten Form dargestellt, es ist der CIC mit nur einer Ebene (*CIC1*). Mathematisch betrachtet liefert dieses Filter exakt das gleiche Ergebnis wie eine gleitende Mittelwertbildung, gefolgt von einer Dezimierung, doch hinsichtlich des Algorithmus ist es wesentlich effizienter.

Das Prinzip ist schnell durchschaut: Wenn beispielsweise N = 10 ist, wird der hundertste gefilterte Abtastwert als Summe der Abtastwerte x_{101} ... x_{110} betrachtet. Statt diese Summe zu berechnen, können wir auch von der Summe der Abtastwerte x_0 ... x_{110} die Summe der Abtastwerte x_0 ... x_{100} subtrahieren. Die Ergebnisse sind absolut identisch.

Natürlich ist es nicht sinnvoll, diesen Prozess für einen einzelnen Abtastwert auszuführen, doch beim Filtern von Signalen ist er gerechtfertigt, weil überflüssige Additionen entfallen. Wie **Bild 2c** zeigt, ist der optimierte Algorithmus ein dreistufiger Prozess. Zuerst muss die Summe der vorausgehenden Abtastwerte berechnet werden. Das ist nicht schwierig, jeder neue Abtastwert wird lediglich einem Addierer zugeführt. Mathematisch betrachtet läuft das Addieren der Abtastwerte auf das Ausführen einer Integration hinaus. Anschließend wird die Dezimierung ausgeführt, was bedeutet, dass nur für einen von D Abtastwerten ein Ausgangswert berechnet wird. Schließlich folgt für jeden zu berechnenden Ausgangswert die Subtraktion der aktuellen Summe von der vorangegangenen Summe über einen FIFO-Puffer (*First In First Out*) aus N / D Elementen. Wenn N = D ist, besteht der FIFO aus nur einem Element. Die Addition oder Subtraktion von zwei zeitverschobenen Größen wird in der Signalverarbeitung *Comb* (deutsch: Kamm) genannt. Mathematisch gesehen hat dies Ähnlichkeit mit einer Differentation. Nun dürfte auch klar sein, weshalb ein solches Filter mit „CIC" (*Cascaded Integrator Comb*) bezeichnet wird. **Bild 3** zeigt die Struktur des CIC1-Filters, das wir hier entworfen haben.

Zur Erinnerung: Das Ergebnis stimmt exakt mit einem einfachen gleitenden Mittelwert überein, die Anzahl der mathematischen Operationen ist jedoch drastisch niedriger. Wir wollen noch einmal unser Beispiel betrachten und einen gleitenden Mittelwert über N = 256 Abtastwerte bilden, gefolgt von einer Dezimierung um den Faktor D = 64. Der CIC-Algorithmus führt mit jedem Eingangswert eine Addition aus, also 44100 Additionen in der Sekunde. Hinzu kommen eine Subtraktion mit jedem Ausgangswert, also 44100 / 64 = 689 Subtraktionen in der Sekunde. Insgesamt müssen 44100 + 689 = **44789** Operationen in der Sekunde ausgeführt werden, während die herkömmliche Berechnung des gleitenden Mittelwerts **176384** Operationen bei exakt gleichem Ergebnis erfordert. Ein wirklich deutlicher Unterschied!

Listing 1.

```
// Länge der Simulation
LENGTH=1024;

// Breite des Fensters
D=8;

// Impulsantwort des Filters (Rechteck)
imp=zeros(1:LENGTH);
imp(1:D)=1/D;

// Zugehörige Frequenzcharakteristik
freqresponse=abs(fft(imp));

// Charakteristik darstellen, logarithmisch und linear
subplot(1,2,1);
plot2d((1:LENGTH/2)/LENGTH,freqresponse(1:$/2));
xtitle('Moving average filter - D=8 (linear scale)');
subplot(1,2,2);
db=20*log10(freqresponse+1e-200);
plot2d((1:LENGTH/2)/LENGTH,db(1:$/2),rect=[0,-25,0.5,0]);
xtitle('Moving average filter - D=8 (dB)');
```

Eigenschaften von CIC-Filtern

Die Frequenzcharakteristik eines solchen CIC1-Filters stimmt natürlich mit der gleitenden Mittelwertbildung überein, denn beide liefern das gleiche Ergebnis. Leider entspricht die Frequenzcharakteristik nicht dem idealen Tiefpass. Bestimmen lässt sich die Charakteristik auf drei Wegen: Mathematisch, durch Simulation oder intuitiv. Wir wollen mit der Mathematik beginnen. Erinnern Sie sich noch

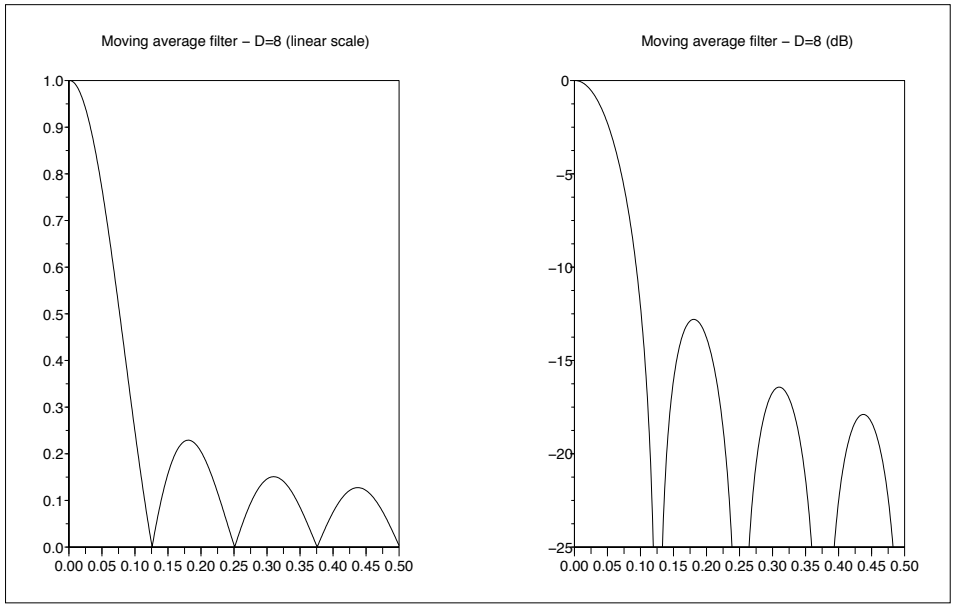


Bild 4. Frequenzcharakteristik eines CIC1-Filters mit Fenster für den Mittelwert aus acht Abtastwerten und der Dezimierung D = 8, dargestellt linear (links) und logarithmisch (rechts). Mathematisch gesehen ist dies die Funktion *sin(x)/x*, sie hat ein Tiefpass-Maximum und mehrere sekundäre Maxima.

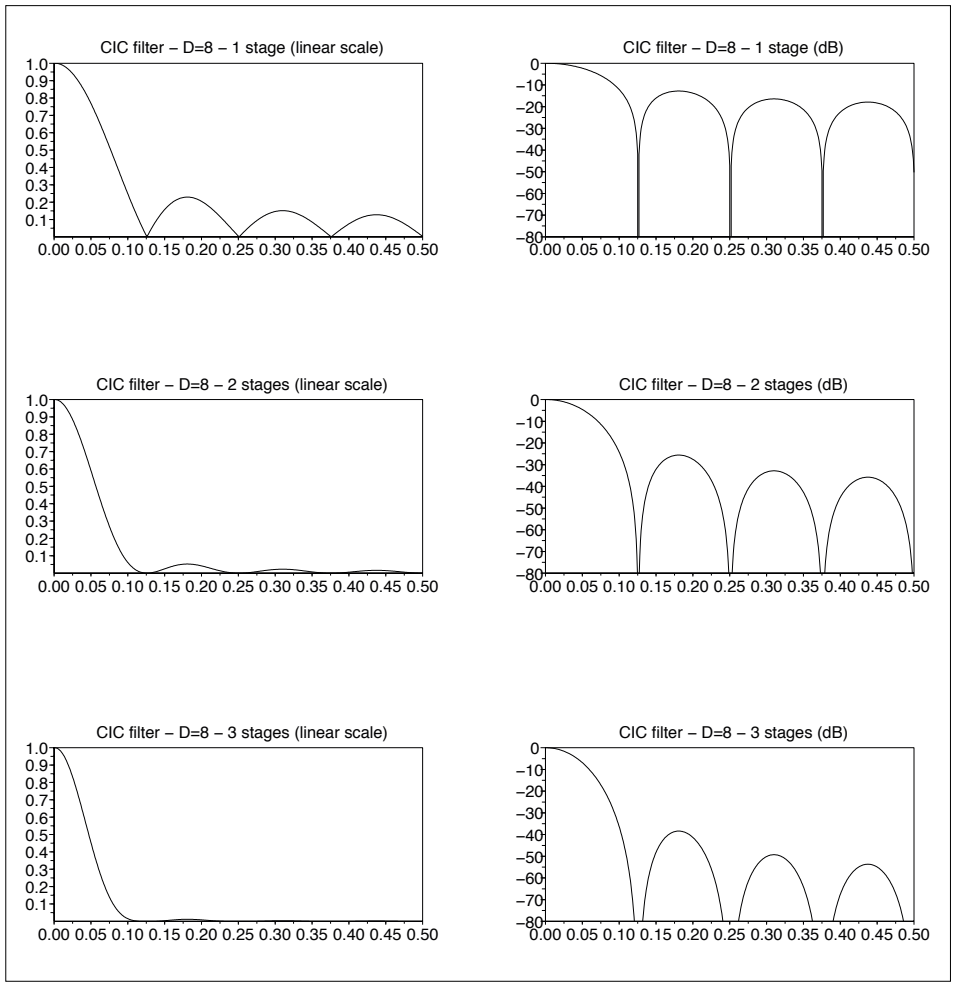


Bild 5. Durch Hintereinanderschalten mehrerer CIC-Filter können die sekundären Maxima reduziert werden, doch dabei nimmt die Anzahl der nötigen Rechenoperationen zu. Das Bild vergleicht CIC-Filter des Typs CIC1 (oben), CIC2 (mittig) und CIC3 (unten).

an meinen Beitrag über die FIR-Filter [1]? Ein Filter mit gleitendem Mittelwert entsteht durch Multiplizieren von N aufeinander folgenden Abtastwerten mit einer Konstanten 1 / N und anschließendem Bilden der Gesamtsumme. Das CIC-Filter ist also ein FIR-Filter mit N *Taps*, wobei sämtliche Koeffizienten gleich 1 / N sind. Die Impulsantwort dieses Filters ist ein Rechteckimpuls mit der Länge N bei konstanter Höhe. Sicher erinnern Sie sich auch daran, dass die Frequenzcharakteristik eines FIR-Filters gleich der Fouriertransformierten seiner Impulsantwort ist. Die Fouriertransformierte eines Rechteckimpulses ist eine in der Signalverarbeitung häufig vorkommende Funktion, sie wird *Sinus Cardinalis* genannt und lautet *sinc(x) = sin(x) / x*. Um die Funktion grafisch darzustellen, haben wir ein Skript für SciLab geschrieben, das die Frequenzcharakteristik eines CIC1-Filters mit D = N = 8 berechnet. Das Skript erklärt sich selbst, auch wenn Sie mit SciLab oder MatLab nicht vertraut sein sollten (**Listing 1**).

Installieren Sie SciLab auf Ihrem PC (falls noch nicht geschehen) und führen Sie das Skript aus. Auf dem Bildschirm erscheint eine Frequenzcharakteristik, wie sie **Bild 4** zeigt. Was können wir dieser Grafik entnehmen? Die Charakteristik verläuft so, wie wir es von einem Tiefpass erwarten. Die Dämpfung nimmt mit steigender Frequenz zu, bis sie bei einer bestimmten Frequenz unendlich hoch wird (Verstärkung ist Null). Für D = 8 liegt dieser Punkt bei der Frequenz, die gleich 1/8 der Abtastfrequenz ist. Das erscheint auch plausibel: Wenn das Eingangssignal eine Sinusfunktion mit exakt dieser Frequenz ist, also seine Periode mit der Länge des berechneten Mittelwerts übereinstimmt, dann muss der Mittelwert immer Null sein!

Wenn jedoch, und das ist die erste schlechte Nachricht, die Frequenz des Eingangssignals diese Frequenz übersteigt, sinkt die Güte des Filters. Dann folgen ein zweites Maximum, bei der die Dämpfung nur 13 dB beträgt, sowie der nächste Nullpunkt, ein drittes Maximum und so weiter. Der Verlauf ist charakteristisch für Filter, die auf gleitenden Mittelwerten basieren.

Die zweite schlechte Nachricht betrifft den Durchlassbereich des CIC1-Filters: Dieser ist absolut nicht eben. Betrachten wir noch einmal Bild 4: Die Dämpfung ist ausschließlich bei der Frequenz 0 Hz (DC) gleich 0 dB, danach steigt sie fortlaufend an. Diese Eigenschaft ist hier deutlich schlechter als beispielsweise bei einem Butterworth-Tiefpass, bei dem der Durchlassbereich viel flacher verläuft. Das ist der Preis, der für die Effizienz des CIC-Filters gezahlt werden muss. Den Filterkonstrukteur kann jedoch nichts davon abhalten, dieses Verhalten durch zusätzliche mathematische Operationen zu korrigieren. Durch Hinzufügen eines FIR-Filters nach der Dezimierung, das im abfallenden Teil der CIC-Charakteristik die exakt inverse Charakteristik aufweist, lässt sich im Durchlassbereich die gewünschte flache Charakteristik realisieren. Ein solches Filter wird mit *Invsinc* bezeichnet. Die gute Nachricht: Da das Korrekturfilter hinter die Dezimierung geschaltet werden kann, hält sich die nötige Rechenleistung des Signalprozessors in Grenzen.

Doch zurück zu den CIC-Filtern. Durch überlegte Wahl der Parameter N und D lässt sich die Charakteristik eines CIC-Filters verbessern, der kuppenartige Verlauf bleibt jedoch erhalten. Die Lösung dieses Problems besteht darin, mehrere CIC1-Filter hintereinander zu schalten. Die Maxima werden zusätzlich abgeschwächt, indem die Mittelwerte der Mittelwerte der Mittelwerte (und so weiter) berechnet werden. **Bild 5** zeigt die Frequenzcharakteristiken bei Reihenschaltung von einem, zwei oder drei CIC1-Filtern mit N = D = 8.

In der Praxis können die CIC-Filter auf geschickte Weise hintereinander geschaltet werden, insbesondere wenn sie mit identischen Parametern N und D arbeiten. Wenn wir beispielsweise zwei CIC1-Filter miteinander kombinieren, reihen sich die Stufen wie folgt aneinander: Integrator, Dezimator, Comb, Integrator, Dezimator, Comb. Da die Operationen linear sind, dürfen wir sie in abweichender Reihenfolge ausführen. Wir erhalten dann die Struktur des CIC2-Filters, das in **Bild 6a** dargestellt ist. Zuerst werden die beiden

Integrationen ausgeführt, daran schließt sich eine einzige Dezimierung an, danach folgen die beiden Comb-Operationen. Nach dem gleichen Prinzip sind auch Filter höherer Ordnung realisierbar, beispielsweise CIC3 oder CIC4.

CIC-Filter können auch eingesetzt werden, um die Abtastfrequenz eines Signals zu erhöhen, auch das kann manchmal sinnvoll sein. Dazu genügt es, den Dezimator durch einen *Interpolator* zu ersetzen und die Folge der übrigen Operationen zu vertauschen: **Bild 6b** zeigt, dass auf den Comb der Interpolator und zuletzt der Integrator folgen. Der Interpolator dupliziert lediglich jeden Abtastwert N mal.

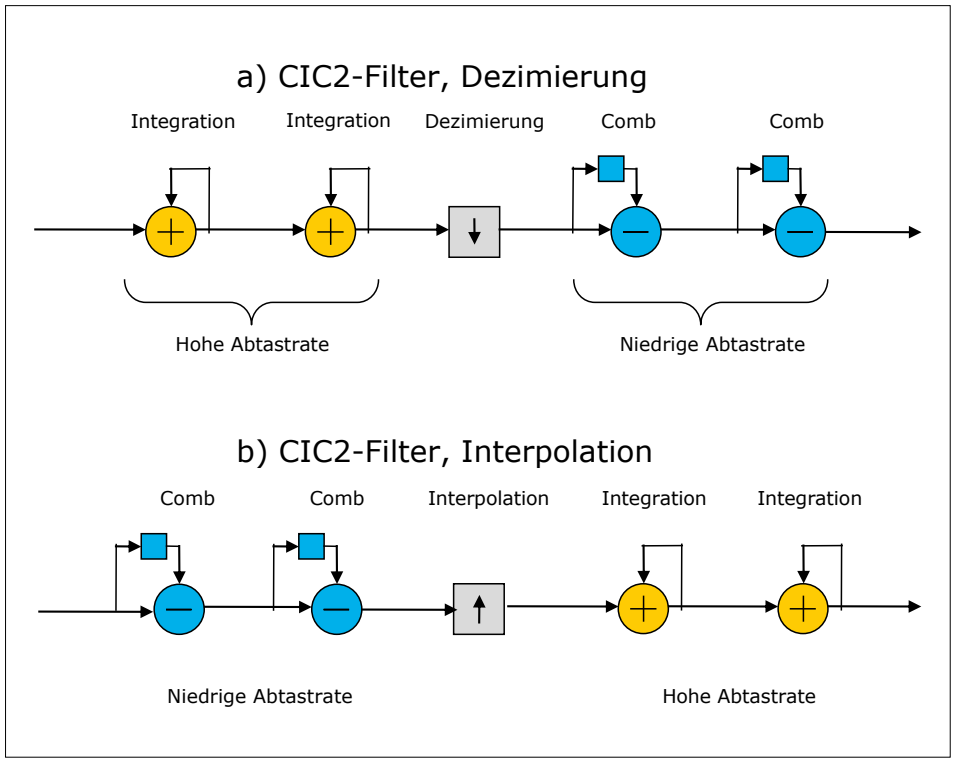


Bild 6. a) Das Umgruppieren der Integrator- und Comb-Stufen von zwei aufeinander folgenden CIC1-Filtern ist eine effiziente Vorgehensweise, um ein CIC2-Filter zu realisieren. b) Umgekehrt kann ein CIC2-Filter dazu dienen, die Abtastrate eines Signals zu erhöhen. Dazu werden die Integrator- und die Comb-Stufen miteinander vertauscht, und der Dezimator wird durch einen Interpolator ersetzt

Umsetzen in die Praxis

Wie können Sie ein CIC-Filter in Ihr nächstes Projekt einbinden? Nun, wahrscheinlich haben Sie dies schon einmal getan, ohne es zu wissen. Viele integrierte Bausteine wenden intern CIC-Filter an, dazu gehören die meisten Komponenten, die mit Oversampling arbeiten: Audio-CODECs, DDS-Frequenzsynthesizer, Sigma-Delta-A/D-Wandler, Funkempfänger mit digitaler Signalverarbeitung im Basisband, und so weiter. Laden Sie sich aus dem Netz als Beispiel die Dokumentation des Highend-Signalprozessors AD6636 herunter (Analog Devices). In diesem Baustein sind sechs Hochgeschwindigkeits-CIC-Filter integriert, die Abtastrate beträgt bis 150 MS/s. Auf die CIC-Filter folgen konfigurierbare FIR-Filter, ihre Aufgabe dürfte nach der Lektüre dieses Beitrags etwas verständlicher sein. Den CIC-Filtern des AD6636 sind digitale Mischer vorgeschaltet, mit denen die Filter auf die gewünschten Frequenzen gezogen werden können.

Den Code für Ihr CIC-Filter können Sie selbst schreiben, beispielsweise in VHDL für einen FPGA sowie in C für einen DSP oder Mikrocontroller. Wie schon erwähnt, sind CIC-Filter genügsam, wenn es um die Rechenleistung geht. Sogar mit kleinen 8-bit-Mikrocontrollen lassen sich CIC-Filter realisieren, die einige Zehntausend Abtastwerte in der Sekunde verarbeiten. Diese Leistung können Sie um eine oder zwei Größenordnungen steigern, indem Sie beispielsweise einen 32-bit-Mikrocontroller Cortex M3 einsetzen, der weniger als einen Euro kostet.

Schlussbetrachtung

Wir haben gesehen, dass CIC-Filter nicht mehr und nicht weniger leisten, als auf geschicktem Weg gleitende Mittelwerte zu berechnen. Wenn die wellige Frequenzcharakteristik tolerierbar ist, können CIC-Filter als digitale Tiefpässe arbeiten. Weil CIC-Filter nur wenig Rechenleistung verlangen, sind sie unter anderem auch zum Wandeln der Abtastrate eines Signals einsetzbar.

In einem folgenden Beitrag aus dieser Reihe werden wir einen Filtertyp betrachten, der bemerkenswerte Eigenschaften aufweist, jedoch etwas schwieriger zu handhaben ist: Das IIR-Filter (*Infinite Impuls Response*). ◀

(150636)gd

Dieser Beitrag erschien auch in Circuit Cellar Nr. 231 vom Oktober 2009.

Links

[1] Digitale Filter einfach erklärt: FIR-Filter, Elektor 12/2016, www.elektormagazine.de/150635

[2] CIC Filter Introduction, Matthew P. Donadio, http://home.mit.bme.hu/~kollar/papers/cic.pdf

[3] Understanding CIC Compensation Filters, Application Note 455, April 2007, Altera Corporation, www.altera.com/literature/an/an455.pdf

[4] Understanding cascaded integrator-comb filters, Richard Lyon, Embedded Systems programming, www.embedded.com/columns/technicalinsights/160400592?_requestid=511057

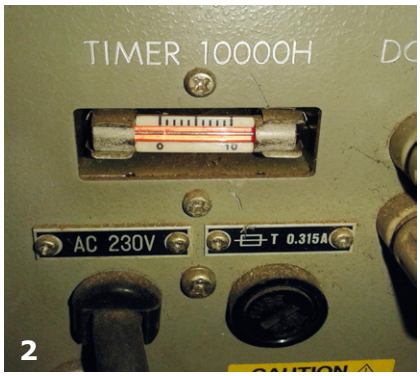
[5] www.scilab.org

Quecksilber-Coulometer

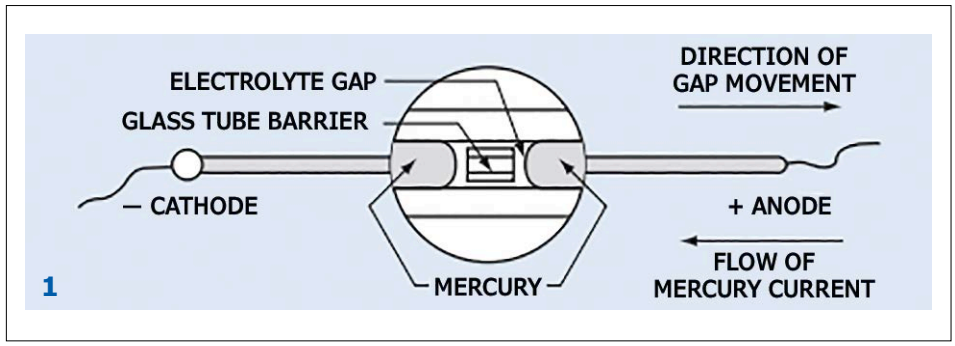
Bemerkenswerte Bauteile

Von **Neil Gruending** (Kanada)

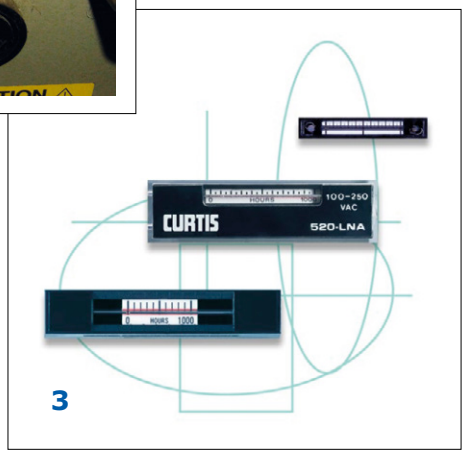
Coulometer sind allgegenwärtig. Sie messen still und leise die Coulombs, die zu und aus Batterie-Packs fließen, damit man die verbleibende Batteriekapazität abschätzen kann. Wie die meisten modernen Messgeräte werden sie als integrierte Schaltung implementiert, Darum war ich fasziniert, als ich einen Thread auf der „TekScopes"-Mailing-Liste über ein Quecksilber-Coulometer entdeckte, das als Countdown-Zähler verwendet wurde, um die Zeit zu schätzen, die ein Messgerät noch mit Energie versorgt sein würde. Schauen wir uns das interessante Innenleben dieser erstaunlichen Messgeräte an!



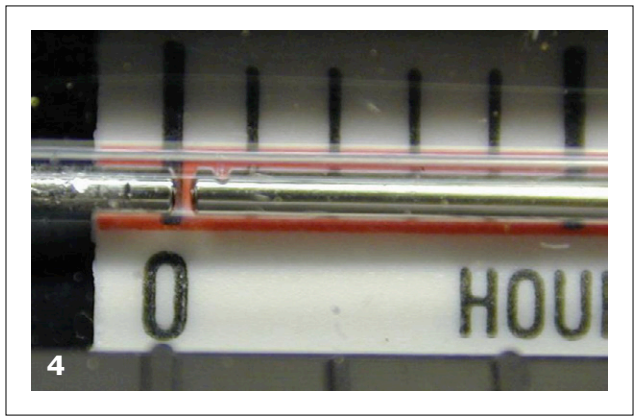
2



1



3


4

Quecksilber-Coulometer wurden ursprünglich 1961 von Curtis Instruments entworfen und bis in das Jahr 2011 hergestellt. Ein solches Coulometer bestand aus einer Glaskapillare, die an jedem Ende mit flüssigem Quecksilber gefüllt wurde, wobei eine geringe Menge Elektrolytlösung zwischen den beiden „Quecksilbern" in einem kleinen Spalt als Indikator verblieb (**Bild 1**). An den Enden des Glasröhrchens wurden Elektroden befestigt, so dass bei einer angelegten Spannung ein Strom fließen würde.

Der Strom veranlasst das Quecksilber auf der Anodenseite, auf elektrochemischen Weg durch den Spalt zur Kathodenseite zu wandern. Die Lücke mit dem Elektrolyt bewegt sich damit Richtung Anode. Fließt der Strom in umgekehrter Richtung, dreht sich natürlich auch der ganze Vorgang um und der Spalt bewegt sich zur anderen Seite. Die Strecke der Bewegung ist proportional zur Strommenge und die Entfernung des Spalts zum Ende der Kapillare gibt Auskunft über den Strom über die Zeit. Dieses Verhalten macht ein Coulomb-Meter aus. Man kann es deshalb auch als Betriebsstundenzähler (meist im Bereich von 1000 Stunden) einsetzen. Durch den einfachen Aufbau ist das Coulometer auch recht leicht, wenig stoßanfällig und dazu auch noch sehr flexibel.

Diese Eigenschaften haben dazu geführt, dass sich Quecksilber-Coulometer als Timer an so interessanten Orten wie im Apollo-Mondlandungsmodul befanden. Nein, befinden, denn die aktuell größte Ansammlung von Coulometern (etwa 20 in jedem Modul) dürfte es noch heute auf dem Mond geben.

Quecksilber-Coulometer haben auch einige Nachteile. Der offensichtlichste ist, dass ein heftiger Schlag oder Stoß das Kapillarrohr beschädigen oder die Elektrolytlösung im Spalt aufwirbeln kann, so dass das Messgerät nicht mehr ordnungsgemäß funktioniert, vor allem, wenn das Quecksilber austritt. Ein weiteres Problem ist, dass, wenn der Elektrolyt an ein Ende des Rohres gelangt, er die Elektrode lösen und ein Leck verursachen kann.

Heutzutage werden (zumindest in der westlichen Welt) keine Quecksilber-Coulometer mehr hergestellt. Sie müssen also schon irgendwo ein Gerät finden (wenn Sie zum Beispiel am Mare Tranquillitatis vorbeikommen), um dieses bemerkenswerte Bauteil zu studieren. Eventuell können Sie aber auch russische Exemplare über Ebay erwerben. ⏮

(160178)

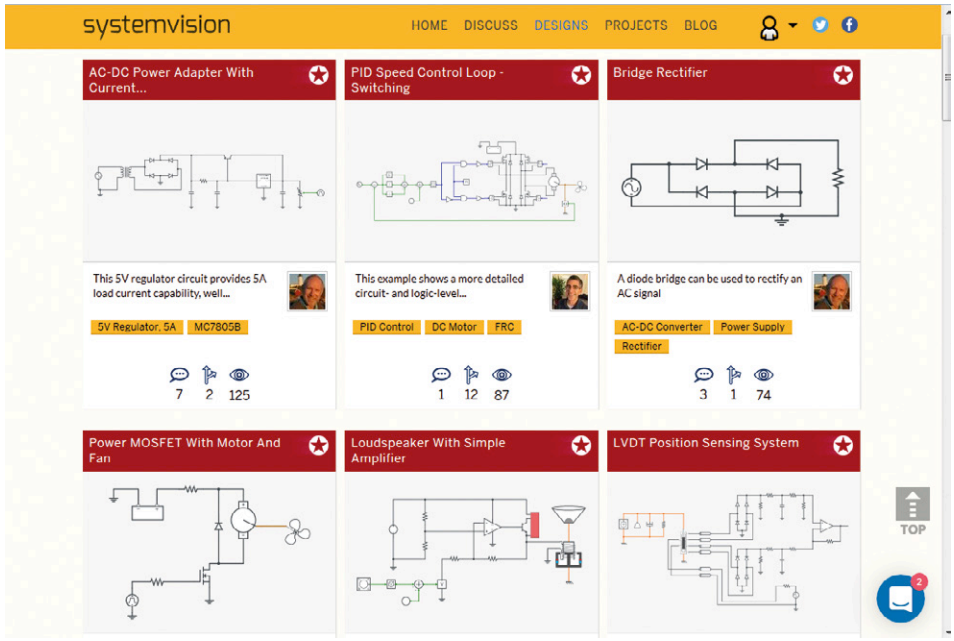
Bildquellen

(1) globalepower; (2) AskElectronics; (3) Octopart; (4) 4hv.

Simulieren mit SystemVision

Kostenlos und web-basiert

Das Simulieren von Schaltungen ist eine wichtige Hilfe, um Fehler in Elektronikschaltungen frühzeitig zu entdecken. Dazu gibt es sehr viele SPICE-basierte Programme, die gewöhnlich recht teuer sind. Kostenlose Versionen gibt es zwar auch, aber die sind meistens ziemlich eingeschränkt. Seit kurzem bietet Mentor Graphics mit SystemVision Cloud eine kostenlose Browser-basierte Simulationsmöglichkeit an. Wir haben uns dieses Programm näher angeschaut.



Von **Tam Hanna** (Slowakei)

Ein alter Spruch besagt, dass die Behebung eines Fehlers umso günstiger ausfällt, desto früher man ihn findet. Insbesondere bei Schaltungen ohne programmierbare Bausteine bietet es sich an, eine grundlegende Verifikation auf dem Computer durchzuführen.
Mentor Graphics bietet mit SystemVision seit längerer Zeit ein umfangreiches Simulationsprogramm für Schaltkreise an, das bisher allerdings ausschließlich kostenpflichtig war. Seit einiger Zeit gibt es mit SystemVision Cloud (**Bild 1**) auch eine kostenlose Basisversion. Der wichtigste Unterschied zur Vollversion besteht darin, dass alle in SystemVision Cloud erstellten Elemente automatisch mit der Allgemeinheit geteilt werden.

Erste Schritte

Für unsere Bedürfnisse mag dies fürs Erste ausreichen: Öffnen Sie die Webseite **www.systemvision.com** und klicken Sie auf den Button *Log In*, um ein Pop-Up-Fenster zum Einloggen zu öffnen.

Zum Zeitpunkt der Drucklegung dieses Artikels akzeptiert SystemVision neben hauseigenen Benutzerkonten auch User, die sich mit ihrem Facebook-, Microsoft-, LinkedIn-, Twitter- und/oder Google-Account beim Dienst anmelden wollen.

Der eigentliche Anmeldeprozess ist dabei von Netzwerk zu Netzwerk unterschiedlich: Im Fall von Twitter erscheint beispielsweise ein Pop-Up-Fenster auf dem Bildschirm, indem Sie SystemVision zum Zugriff auf Ihr Konto autorisieren müssen.

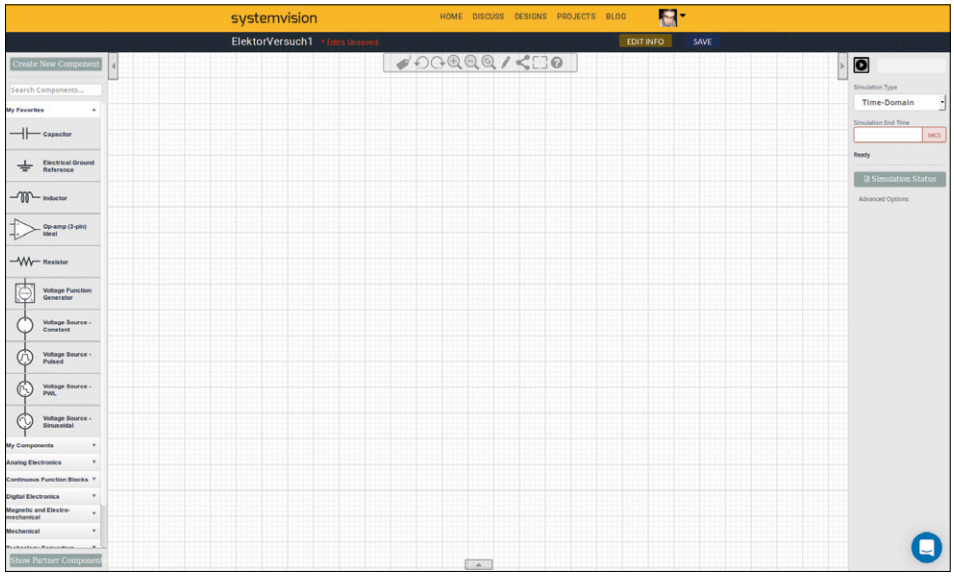


Bild 1. Die Arbeitsoberfläche von SystemVision.

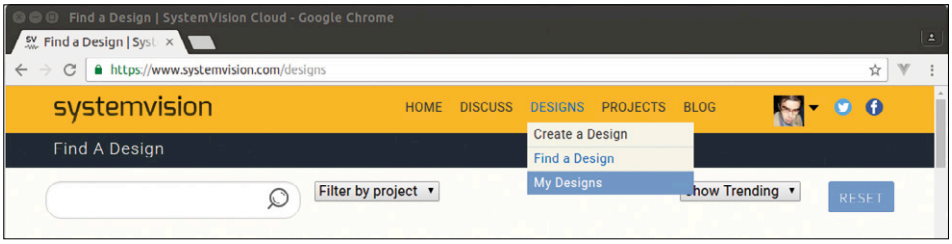


Bild 2. Das Kontextmenü von SystemVision ist sehr wichtig.

Durch den sehr starken sozialen Fokus von SystemVision ist das Anlegen eines neuen Projekts etwas kompliziert. Klicken Sie auf die gelbe Task-Leiste auf der Oberseite des Bildschirms und wählen Sie die Option *Designs* ➞ *Create a Design* aus (**Bild 2**).

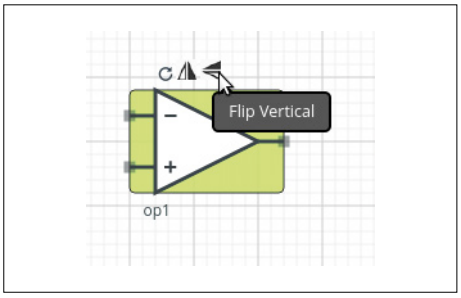


Bild 3. Komponenten lassen sich nach Belieben drehen und wenden.

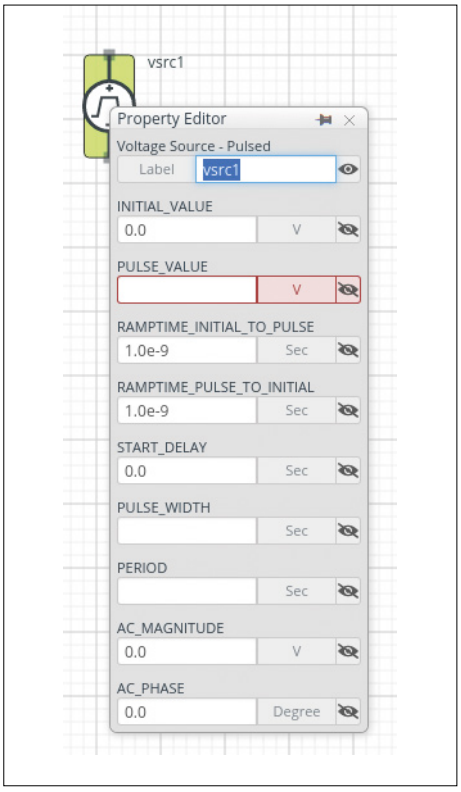


Bild 4. Das Fenster zum Eingeben von Attributen ist gut versteckt.

SystemVision erlaubt beim nächsten Schritt die Auswahl eines schon vorhandenen Designs. Wer mit einem komplett leeren Zeichenbrett anfangen möchte, klickt stattdessen auf *My Designs*. Im daraufhin erscheinenden Bildschirm klicken Sie auf die Funktion *Create*. SystemVision startet dann die eigentliche Design-Ansicht, die den Assistenten zum Öffnen eines neuen Designs auf den Bildschirm holt.

Das Programm lässt sich beim Laden einige Zeit: Je nach Systemkonfiguration kann es in Google Chrome oder Firefox schneller gehen als im klassischen Internet Explorer. Wenn Sie das Design frisch generieren, müssen Sie einen Titel und die Sichtbarkeit eingeben. Wer im Feld *Visibility* den Wert „Show only to me“ auswählt, erzeugt ein privates Design.
Das eigentliche Benutzerinterface von SystemVision fällt einfach aus. Auf der linken Seite findet sich eine Liste der diversen in SystemVision enthaltenen Komponenten. Diese lassen sich per Drag and Drop auf dem Arbeitsbereich platzieren. Die gerade selektierte Komponente bekommt in grau Optionen zum Drehen und Spiegeln eingeschrieben (**Bild 3**). Nutzen Sie diese, um die Lage der Komponente an die Bedürfnisse ihres Schaltkreises anzupassen.

Drahtverbindungen lassen sich dann per Drag and Drop zwischen den Toren der Komponenten setzen. Legen Sie den Mauszeiger dazu im ersten Schritt auf eines der Tore - er verwandelt sich dann in ein Fadenkreuz. Ziehen Sie im nächsten Schritt eine Linie zum Ziel und legen Sie sie dort ab, um die Verbindung komplett zu konfigurieren.

Wir wollen hier als erstes Beispiel einen primitiven RC-Schaltkreis realisieren, der die klassische langsam ansteigende Spannung nach dem Einschalten der Stromversorgung demonstriert.
Ziehen Sie die benötigten Elemente per Drag and Drop auf die Anzeigefläche und verbinden Sie sie wie von Fritzing und Co. gewohnt. Nun stellt sich die Frage, wie man in SystemVision die Werte der einzelnen Komponenten festlegt. Der Trick dazu ist das Anklicken des Labels, was mit dem in der Abbildung gezeigten Dialog belohnt wird (**Bild 4**). Geben Sie dort die gewünschten Parameter ein und klicken Sie an eine Stelle außerhalb des Fensters, um die Werte in den Speicher zu übernehmen.

Erweitern Sie unseren Schaltkreis wie in **Bild 5** gezeigt, um die Voraussetzungen für eine erfolgreiche Simulation des RC-Glieds zu schaffen. Wichtig ist in diesem Zusammenhang, dass SystemVision Einheiten von Haus aus nur in Ohm, Farad usw. entgegennimmt. Die Größenordnung verändern kann man mit den folgenden Buchstabenkombinationen, die hinter dem jeweiligen numerischen Wert eingegeben werden können:

- f
- p
- n
- u
- m
- K
- Meg
- G
- T

Aufmerksame Leser fragen sich an dieser Stelle, wieso die Schaltung statt einer konstanten Spannungsquelle eine gepulste Spannungsquelle einsetzt. Dieses auf den ersten Blick seltsame Verhalten liegt an einem kleinen Bug: Wer ein RC-Glied aus einer konstanten Spannungsquelle speist, setzt einen Teil der Simulationsumgebung außer Gefecht und bekommt statt charakteristischer Wellenformen nur konstante Spannungen.

Simulatives!

Vor dem eigentlichen Anstoßen der Simulation ist noch die Deklaration eines Erdsymbols notwendig: Die hinter SystemVision stehende Simulationsengine funktioniert nur dann, wenn ihr ein Basispotenzial zugewiesen wird. Klicken Sie dann in der *Emulation Toolbar* auf der rechten Seite des Bildschirms in das Feld

Simulation End Time, und geben Sie fünf Sekunden ein. Klicken Sie dann auf das Pfeilsymbol ganz oben, um eine Simulation des Typs Time-Domain anzustoßen. SystemVision ist auch zur Simulation im Frequenzbereich befähigt – ein Feature, auf das wir an dieser Stelle schon aus Platzgründen nicht weiter eingehen werden.

Nach der auf dem Rechner von Mentor Graphics erfolgenden Simulation blendet das Programm das in **Bild 6** gezeigte Resultatfenster ein, das die diversen, während der Simulation angefallenen Größen auflistet. Eventuelle Fehler im Schaltungsmodell scheinen in Form von roten Bubbles am Bildschirm auf, die Fehlermeldungen sind im Allgemeinen leicht verständlich.

Wer sich das permanente Durchgraben der Ergebnisordner ersparen möchte, kann das Bleistiftsymbol zum Anlegen eines *Watchers* (**Bild 7**) nutzen. Es handelt sich dabei um ein über der Schaltung schwebendes Element, das die Ergebnisse des aktuellen Simulationsdurchlaufs beschreibt.

Zum Zeitpunkt der Drucklegung weist SystemVision im Bereich des Simulationssystems einen ärgerlichen Bug auf: Wer eine zweite Simulation anstößt, bekommt im Simulationsfenster einen weiteren Ordner angelegt. Aus diesem Grund ist es empfehlenswert, nicht mehr benötigte Diagramm- oder Simulations-Elemente mit der rechten Maustaste anzuklicken und über die Menüoptionen zu beseitigen. Ein weiteres kleines Ärgernis ist das Speichern der Änderungen: SystemVision übernimmt die im Browserfenster gemachten Modifikationen der Schaltung nur dann in den Speicher, wenn Sie auf den Save-Button klicken, der in der weiter oben erwähnten gelben Toolbar liegt.

> Das wichtigste Feature ist die Möglichkeit zum Teilen von Designs mit Freunden und Kollegen.

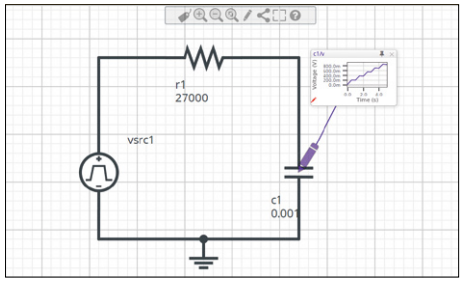


Bild 5. Der RC-Kreis ist zur Simulation bereit.

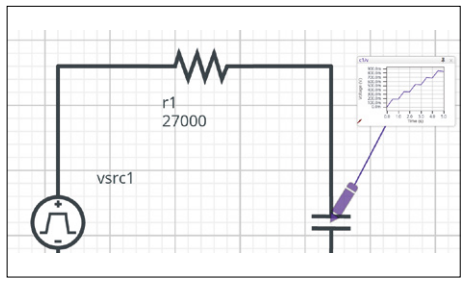


Bild 7. Mit dem Bleistiftsymbol kann man einen sogenannten Watcher anlegen.

Soziales und mehr

Auch wenn die Simulation in SystemVision hochinteressant ist: Das wichtigste Feature ist die Möglichkeit zum Teilen von Designs mit Freunden und Kollegen. Unser Design lässt sich durch Anklicken des „Share Design"-Buttons freigeben – SystemVision blendet daraufhin eine URL (z.B. http://sysvis.io/smKBn9) ein, die mit Freunden und Kollegen geteilt werden kann.

Die kostenlose Basisversion von SystemVision erlaubt bis zu fünf private Designs. Mehr gibt es nur für Besitzer der Professional-Variante (Sie können ein nicht mehr benötigtes Design aber löschen und damit unter der maximalen Anzahl von fünf Stück bleiben; oder auch ein Design freigeben). Leider führt der Klick auf den „Upgrade Account"-Button nur zu einer E-Mail-Adresse: Preisinformationen suchen Sie im Internet vergebens. Wer auf der Suche nach Inspiration ist, kann in der Rubrik „Find a Design" die von anderen Entwicklern bereitgestellten Schaltungen betrachten. Dies ist insbesondere dann hilfreich, wenn man mehr über ein bestimmtes Bauteil erfahren möchte oder sich Hinweise zum effizienten Aufbau von Simulationen erhofft.

Fazit

Auch wenn SystemVision wegen der komplett Cloud-basierten Bedienung auf den ersten Blick ein wenig kompliziert ist: Das Produkt bietet einen immensen Funktionsumfang, der in einem einzelnen Artikel bestenfalls angerissen werden kann. Wer sich schon immer für die Simulation von Elektronikschaltungen interessiert hat, sollte dem Produkt definitiv eine Chance geben - es ist PSpice insbesondere in Sachen Usability meilenweit voraus. ◀

(160203)

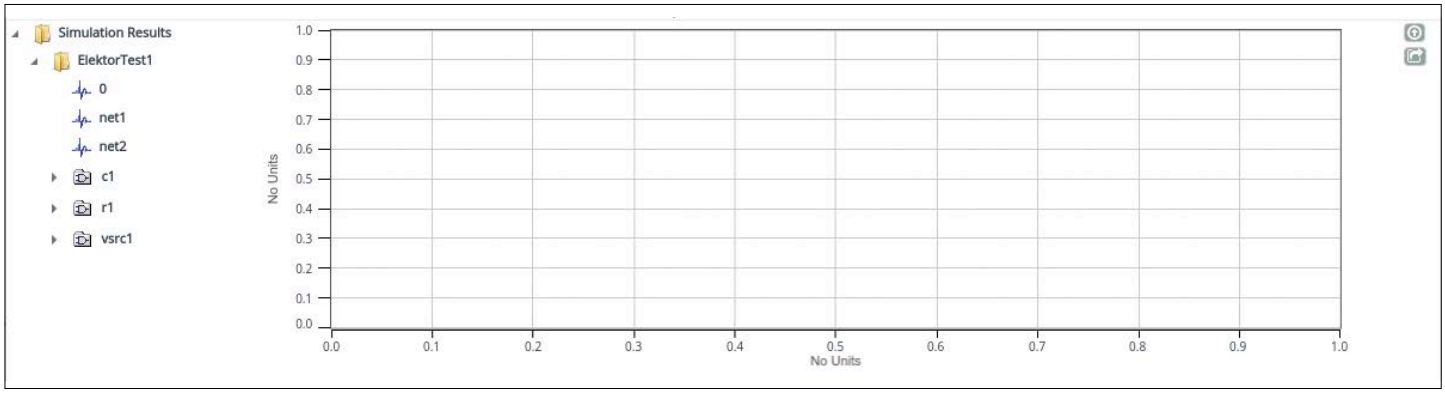


Bild 6. Die Ergebnisse der Simulation sind eingetroffen.

Tipps und Tricks

Von Lesern für Leser

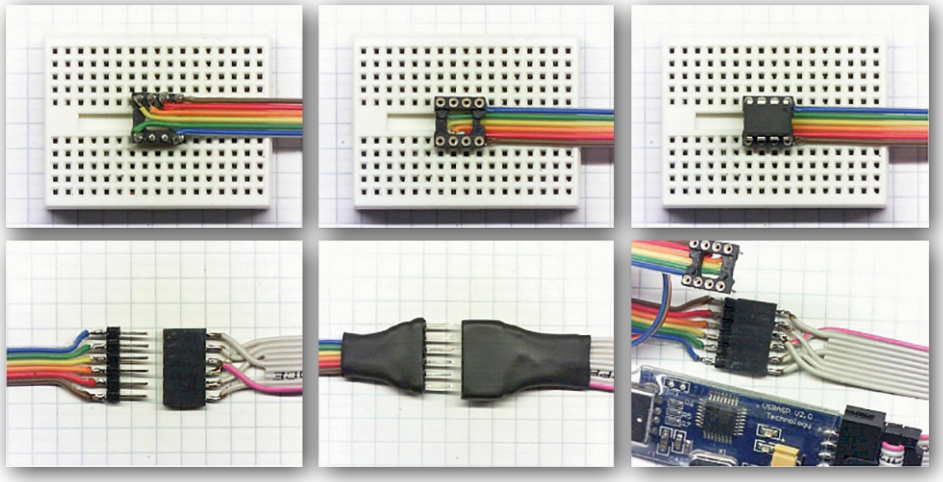
Hier kommt wieder eine clevere Lösung, die das Elektronikerleben leichter macht.

Programmieradapter für USBasp

Von Antonello della Pia

Ich finde es toll, Ideen mit der Elektronikergemeinde zu teilen (und dafür auch noch 40 Euro einzustecken). Einer meiner liebsten Mikrocontroller ist der ATtiny85, den ich gerne mit dem einfachen und preiswerten USBasp programmiere. Leider ist es bei einem Probeaufbau auf einem Steckbrett nicht so einfach, den Programmer an die Pins der ICSP-Schnittstelle des ATtiny anzuschließen.

Um einen passenden Adapter für den USBasp zu konstruieren, bin ich folgendermaßen vorgegangen. Ich habe eine gewöhnliche achtpolige IC-Fassung mit gedrehten runden Löchern und eine 1x6-polige Stift-/Buchsenleiste genommen, dazu habe ich von einem alten, neunpoligen Computer-Flachbandkabel (ehemals für die serielle Schittstelle) sechs Adern abgetrennt. Die sechs Adern führen die Signale MISO, MOSI, SCK, RESET, GND und VCC (wenn Sie die Schaltung aus dem Programmer versorgen möchten); sie werden an die entsprechenden Pins der Fassung gelötet. Auf der anderen Seite des Kabels wird die Stiftleiste angebracht. Wozu diese? Da man die Adern des Kabels unter der Fassung nicht auch noch kreuzen kann, habe ich ein zweites Kabel angefertigt, das die Leitungen passend für den USBasp dreht, und mit einer Buchsenleiste ausgestattet ist. Beachten Sie die unterschiedliche Pinnummerierung bei Fassung und USBasp-Header (Tabellen 1 und 2)!



Das Foto macht klar, wie alles zusammengehört. Ich habe sogar noch einen zweiten Nutzen für das Adapter entdeckt! Wenn man eine sehr kleine Platine entwirft und keinen Platz für einen ICSP-Verbinder verschwenden will, kann man den Adapter einfach zwischen den ATtiny und die Controllerfassung auf der Platine stecken und den Controller auf diese Weise programmieren.

Ich denke, das Prinzip lässt sich ohne weiteres an andere Programmer und Mikrocontroller anpassen. ◀

(160277)

Tabelle 1. Pinbelegung der Programmierschnittstelle (ICSP) beim ATtiny85.

Signal	Pin	Pin	Signal
RESET	1	8	VCC
nc	2	7	SCK
nc	3	6	MISO
GND	4	5	MOSI

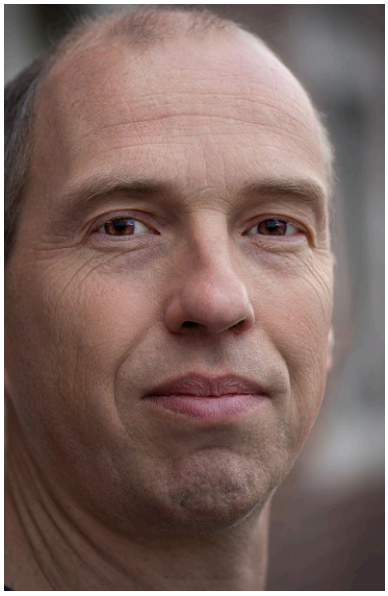
Tabelle 2. Anschlussbelegung des USBasp.

Signal	Pin	Pin	Signal
MOSI	1	2	VCC
GND	3	4	TXD
RESET	5	6	RXD
SCK	7	8	GND
MISO	9	10	GND

Sie haben selbst eine clevere Lösung für etwas wirklich Fummeliges? Wenden ein Bauteil oder Werkzeug auf ungewöhnliche Weise an? Haben eine Idee, wie man ein Problem einfacher oder besser angehen könnte, als das bisher gelöst wurde? Schreiben Sie uns – für jeden Tipp, den wir veröffentlichen, loben wir 40 Euro aus!

Willkommen bei DESIGN

Von **Clemens Valens,** Elektor-Labor



Heinrich Rudolf Hertz
(1857 – 1894)

Hamburg wurde durch zwei berühmte Dinge weltbekannt: das Hamburger-Sandwich, besser bekannt als „Der Burger“, und das Hertz, die SI-Einheit der Frequenz. Beide Berühmtheiten stammen aus der zweiten Hälfte des 19. Jahrhunderts, aber es ist vor allem das Hertz, dem wir die Aufmerksamkeit in dieser Kolumne widmen. Also, wer war der Mann, der die Frequenz erfunden hat?

Heinrich Rudolf Hertz wurde in eine reiche Familie hineingeboren. Sein Vater war Rechtsanwalt und irgendwann sogar Senator von Hamburg. Heinrich zeigte schnell ein großes Interesse an Wissenschaft und Sprachen und besuchte gute Universitäten. Er promovierte in Berlin, wo er bei den allen Elektronikern bekannten wissenschaftlichen Schwergewichten Hermann von Helmholtz und Gustav Kirchhoff studierte. Heinrich hat natürlich die Frequenz nicht erfunden. Er arbeitete mit dem Elektromagnetismus, als Helmholtz ihm vorschlug, die Maxwell'sche Theorie zu beweisen. Obwohl er anfangs nicht glaubte, dass dies machbar sei, kehrte er Jahre später zu diesem Thema zurück, als er versehentlich die Werkzeuge erfand, die es doch möglich machten. Nach dem Beweis der Maxwell'schen Theorie kümmerte er sich nicht mehr darum, weil er nicht an eine praktische Verwendungsmöglichkeit glaubte.

Heinrich war beileibe kein Fachidiot, er schuf auch Grundlagen der Kontaktmechanik, indem er das Kontaktproblem zweier elastischer Körper mit gekrümmten Oberflächen löste. Mit seiner Beobachtung, dass ein geladenes Objekt seine Ladung schneller verliert, wenn es durch ultraviolette Strahlung beleuchtet wurde, half er mit, den photoelektrischen Effekt zu beschreiben, ja er legte damit sogar die Basis für die Erforschung der Röntgenstrahlen. Leider starb Heinrich viel zu jung an einer seltenen Krankheit, die erst 40 Jahre später offiziell entdeckt wurde.

Dennoch gelang es Heinrich, sein Genie an seine Nachkommen weiterzugeben. Seine jüngste Tochter Mathilde wurde eine weltberühmte Biologin und Psychologin, sein Neffe Gustav Ludwig gewann 1925 den Nobelpreis für Physik.

Die SI-Einheit für Frequenz wurde das Hertz (Hz) erst im Jahr 1960, als es den „cycle per second“ (cps) ersetzte. Ein Hertz ist als 1/Sekunde oder s^{-1} definiert, genau wie das Becquerel (Bq). Bei einem Hertz finden die Ereignisse aber genau im Sekundentakt statt, bei einem Becquerel nur gemittelt.

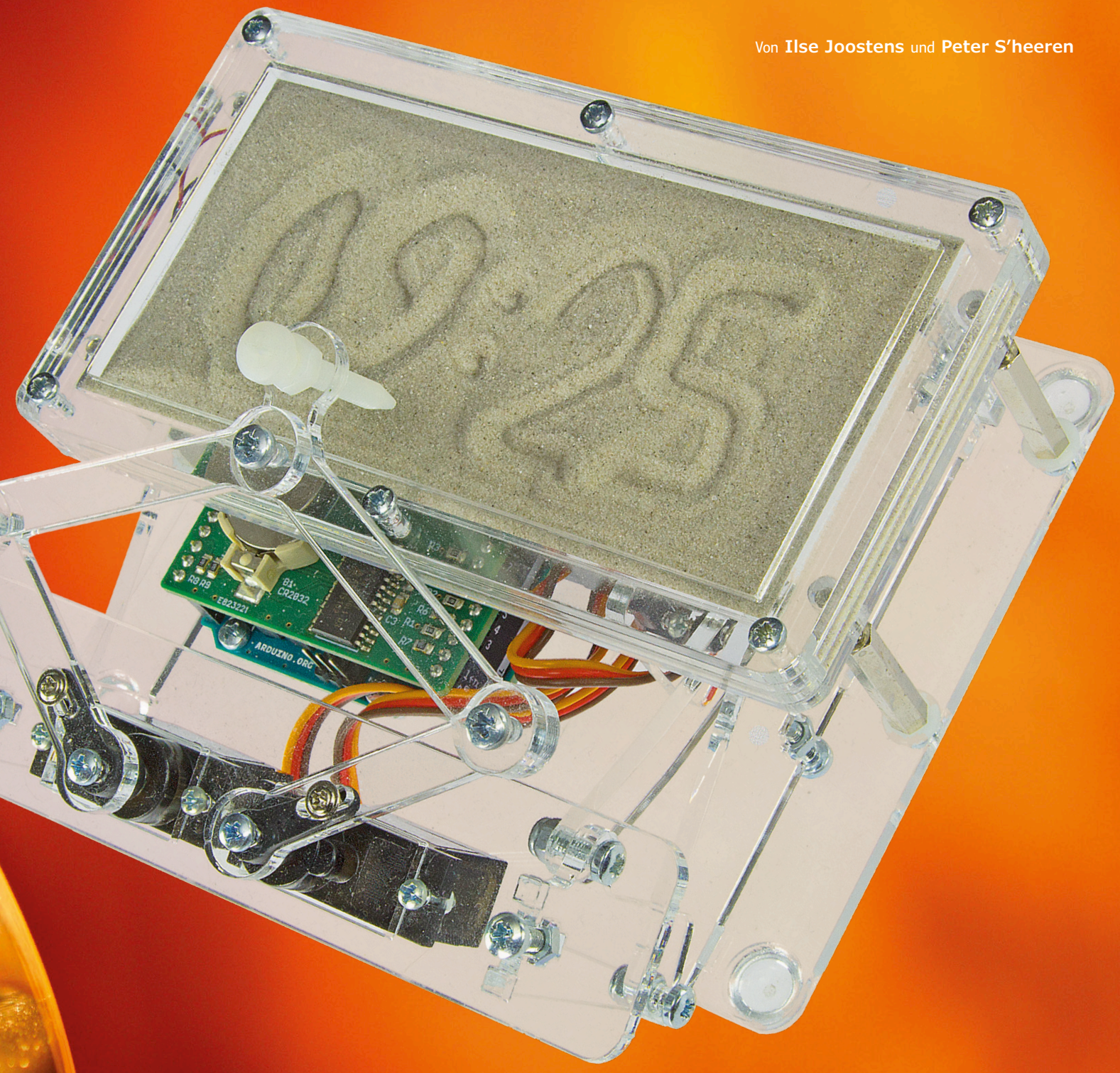
Wo Joseph Henry ein nach ihm benanntes Gebirge besitzt, kann Heinrich auf einen Krater auf der dunklen Seite des Mondes verweisen. Ich habe im Internet nach einem Rezept für einen Hertz-Burger gesucht, der die beiden Berühmtheiten Hamburgs vereint. Aber vergeblich...

(160247)

Sanduhr

Eye-catcher par excellence

Von Ilse Joostens und Peter S'heeren



Das supercoole Gadget, in dem ein Arduino Uno tickt, schreibt die Zeit mit mehreren Servomotoren und einer Pantograf-Mechanik auf eine sandbedeckte Fläche. Nach vorgegebener Zeit glätten zwei Rüttelmotoren die Sandoberfläche, und alles beginnt von vorn. Die Software setzt nicht nur die Zeit in den Sand, die Bewegungen des Schreibstifts lassen sich auch mit einfachen Kommandos steuern.

Die Idee für dieses Projekt stammt von unseren Kollegen, die bei der deutschen Ausgabe des *Make:* Magazins tätig sind. Sie waren inspiriert durch die *Plotclock* des *FabLab Nürnberg*. Bei einem ersten Hardware-Meeting mit den Kollegen von *Make:* wurden Projekte gesichtet, die geeignet sind, im Elektor-Labor angepasst und bis zur Nachbaureife einschließlich eines Bauteilesatzes weiterentwickelt zu werden. Das erste Projekt, für das die Entscheidung positiv ausfiel, war die hier und jetzt präsentierte Sanduhr.

Zuerst wurden die mechanischen Komponenten angepasst, damit sie für einen Bauteilesatz und eine unkomplizierte Montage tauglich sind. Für die periphere Elektronik wurde ein Shield entworfen, und schließlich wurde die originale Software überarbeitet, um sie übersichtlicher und anwendungsfreundlicher zu gestalten und noch einige Funktionen hinzuzufügen. Das alles steckt in dem Projekt, das wir hier nicht ganz ohne Stolz präsentieren.

Einen Bauteilesatz einschließlich eines Arduino Uno und eines vormontierten Shields liefert der Elektor-Shop (**Bild 1**). Wer über einen Lasercutter verfügt und SMD-Lötarbeiten nicht scheut, findet die CAD- und Gerber-Dateien zum Herunterladen auf der Projektseite [1].

Die mechanischen Bauteile unserer Sanduhr bestehen zum größten Teil aus extrudiertem PMMA, auch bekannt als Acrylglas sowie unter dem Markennamen Plexiglas. Dieses Material ist nicht nur leicht erhältlich, sondern eignet sich auch hervorragend für die Bearbeitung mit einem Lasercutter. Die guten thermoplastischen Eigenschaften sind natürlich ebenso willkommen.

Wie bei der originalen *Plotclock* kommen auch hier drei Servos aus dem Modellbau zum Einsatz. Da der Antrieb mit geringen Kräften auskommt, werden an die Servomotoren keine allzu hohen Anforderungen gestellt. Wegen der einfachen Montage der Antriebsachse und des geringen mechanischen Spiels fiel unsere Wahl auf Servomotoren mit metallenen Zahnrädern.

Das Sandbett wird von zwei Schwingmotoren durch Rütteln geglättet, diese Motoren sind mit Kabelschellen aus Kunststoff im Winkel von 45° unter dem Sandbett angebracht.

Pantograf-Mechanik

Zu den Hauptelementen der Sanduhr-Mechanik gehören der linke und der rechte Servomotor sowie die PMMA-Arme, die den Schreibstift bewegen. Die Anordnung bildet einen so genannten *Pantografen*, das Prinzip entspricht einem in früheren Zeiten gebräuchlichen Zeichengerät gleichen Namens, auch *Storchenschnabel* genannt.

Eine in ihrer Mechanik vergleichbare Konstruktion ist die Unterschriftenmaschine, die mit dem Mechanismus des Pantografen Unterschriften reproduzieren kann. Bis zum heutigen Tag sind solche Maschinen beispielsweise bei Politikern und anderen Persönlichkeiten in Gebrauch, die viele Unterschriften leisten müssen. Auch bei personifizierten und unterschriebenen Werbeschreiben werden manchmal solche Maschinen eingesetzt, da die Unterschrift einer Unterschriftenmaschine persönlicher erscheint als eine gedruckte Unterschrift. Einige amerikanische Präsidenten nutzten intensiv Unterschriftenmaschinen, darunter war auch Barack Obama. Er ließ damit Gesetze unterschreiben, als er sich zu einem Staatsbesuch in Frankreich aufhielt oder als er seinen Urlaub auf Hawaii verbrachte.

Nebenstehend haben wir unter *Inverse Kinematik* die mathematischen Hintergründe zusammengefasst, die auch für unsere Sanduhr Gültigkeit haben. Sie sind die Grundlage der Firmware, mit der unsere Sanduhr läuft.

Gerüttelter Sand

Nicht die höhere Mathematik war die schwierigste Hürde dieses Projekts, son-

Bild 1. Der Bauteilesatz umfasst sämtliche Einzelteile einschließlich Steckernetzteil. Der Arduino muss mit der frei herunterladbaren Software programmiert werden.

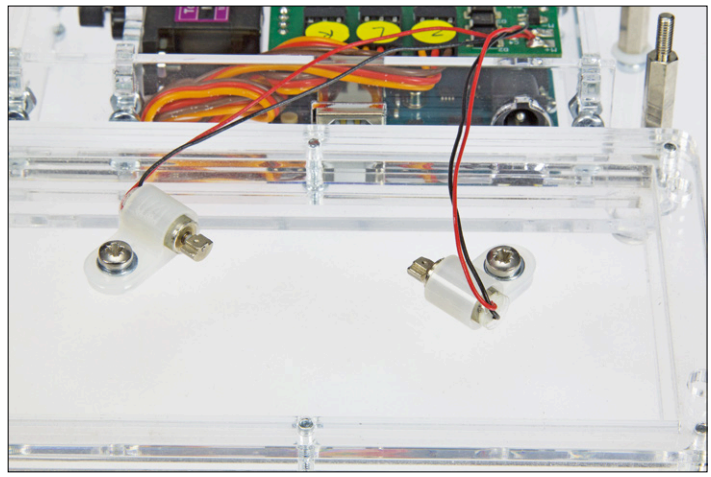


Bild 2. Die Positionierung der Rüttelmotoren unter dem Sandbett ist für das Glätten des Sandes (Löschen der Zeitanzeige) entscheidend.

dern das Glätten der Sandfläche. Ungezählte Prototypen wurden gebaut und getestet, bevor wir mit dem Ergebnis zufrieden waren. Die Vibrationsmotoren sollen nicht die gesamte Uhr durchrütteln, sondern die Rüttelenergie muss sich auf die Sandfläche konzentrieren. Deshalb wird das Sandbett von vier elastischen Stützen getragen, so dass es horizontal in dem nötigen Maß beweglich ist.

Die kniffligste Aufgabe bestand darin, das Sandbett in etwa fünf Sekunden so zu glätten, dass die Zeitinformation neu geschrieben werden kann und stets gut lesbar ist. Der Sand darf nicht zum Anhäufen kleiner oder auch größerer Hügel neigen. Nach Versuchen mit diversen Konstruktionen und unterschiedlichen Rüttelmotoren war klar, dass nur ein Exemplar eines Rüttelmotors diese Aufgabe nicht bewältigen kann. Das beste Ergebnis lieferten zwei 6-mm-Schwingmotoren, die auf der Unterseite des Sandbetts montiert waren. Dabei ist die Montageposition von ausschlaggebender Bedeutung. Die Motoren, so stellte sich heraus, müssen im Winkel von 45° zum Rand der Grundplatte angebracht werden (**Bild 2**).

Auch die Beschaffenheit des Sands ist wichtig. Nicht nur die Korngröße (grober oder feiner Sand), sondern auch die Form der Sandkörner (scharfkantig oder rund) hat maßgeblichen Einfluss. Gute Ergebnisse haben wir mit feinem weißem Sand erzielt (so genannter Silbersand), der eine Korngröße von 0,1 mm bis 0,3 mm hat. Wegen der Konstruktion des Sandbetts raten wir von extrem feinem Sand ab.

Damit der Sand nicht aus dem Sandbett fällt, wird das Sandbett von einem schmalen, überhängenden Rand umrahmt.

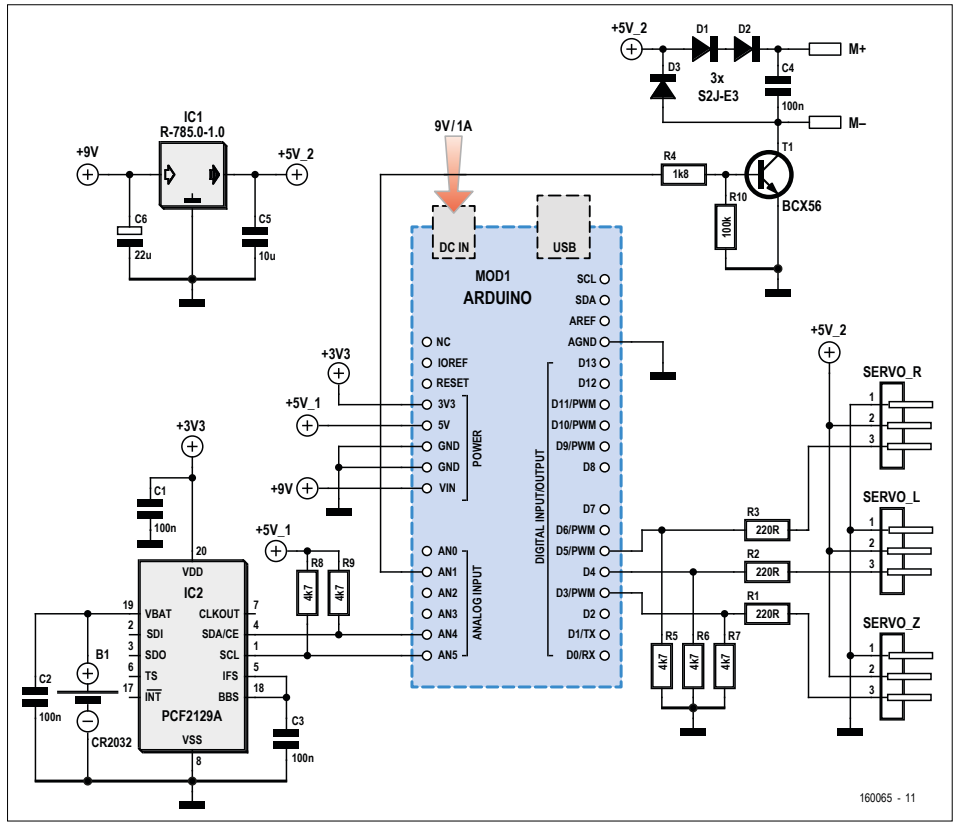


Bild 3. Schaltung des Shields, das für die Sanduhr entworfen wurde. Dazu gehören ein DC/DC-Wandler für die Motorspannungen, ein Realtime-Clock-IC, ein Treiber für die Rüttelmotoren und diverse Steckverbinder für die Servos.

> Die Sanduhr ist auch über ASCII-Kommandos steuerbar.

Elektronik

Wie schon in der Einleitung erwähnt, tickt in unserer Sanduhr ein Arduino Uno. Der Arduino Uno besitzt genügend Rechenpower, trotz der vielen Rechenschritte, die in der Rubrik *Inverse Kinematik* zusammengefasst sind. Außerdem sind weitere Komponenten beteiligt, zur Stromversorgung der Servomotoren, zum Steuern der rüttelnden Motoren und zum Weiterlaufen der Zeit, falls die Netzspannung ausfällt. **Bild 3** zeigt schematisch den elektronischen Teil der Sanduhr.

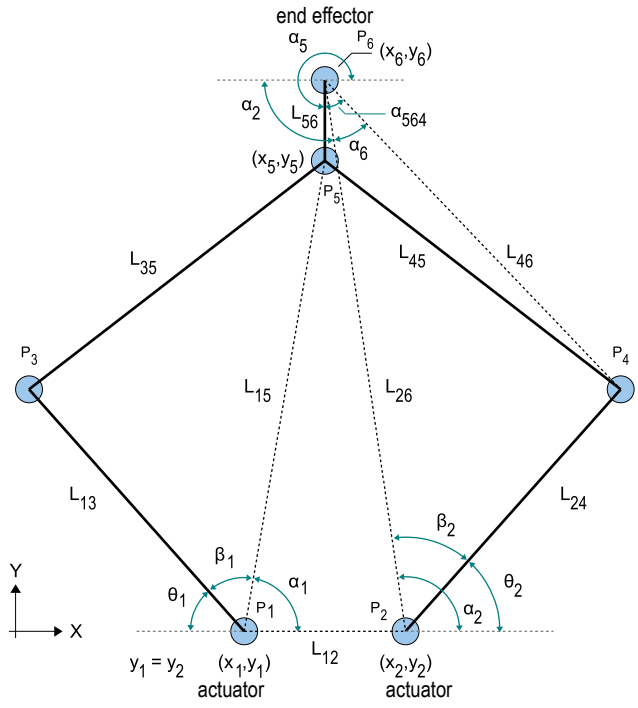
Der Nachbau wird dadurch vereinfacht, dass wir die externe Elektronik auf einem Arduino-Shield untergebracht haben. Die Platine (**Bild 4**) ist überwiegend mit SMDs bestückt. Eventuelle Lötprobleme werden dadurch umgangen, dass die Platine vormontiert zum Bauteilesatz des Elektor-Shops gehört. Nur die Kontaktleisten und der DC/DC-Wandler müssen noch montiert werden.

Mit Strom wird die Sanduhr von einem handelsüblichen, schaltenden Steckernetzteil versorgt, das 9...12 V liefert. Die Spannung 5 V auf dem Arduino-Board ist ungeeignet, da der dort befindliche lineare Spannungsregler nicht nur einen niedrigen Wirkungsgrad hat, sondern auch nur bis 800 mA belastbar ist. Beim Schreiben in den Sand würde einige Energie als Verlustwärme verlorengehen. Mit einer USB-Verbindung zwischen Computer und Sanduhr bestünde außerdem die Gefahr, dass die USB-Versorgungsspannung zusammenbricht, falls das Steckernetzteil versehentlich nicht eingesteckt ist.

Aus den genannten Gründen werden die Motoren unabhängig vom Arduino über einen DC/DC-Wandler versorgt. Der Wandler (IC1), der die Bauform eines Spannungsreglers der Typenreihe 78xx hat, liefert bei Eingangsspannungen 9...12 V die Ausgangsspannung 5 V bis 1 A, mit einem Wirkungsgrad größer 90 %. Die Rüttelmotoren werden über einen NPN-Transistor (T1) von einer I/O-Leitung des Arduino gesteuert. Die Dioden D1 und D2 reduzieren zusammen mit dem Spannungsabfall am Transistor die Spannung der Motoren, da ihre Nennspannung laut Datenblatt nur 3 V beträgt.

Drei I/O-Leitungen des Arduino steuern die Servomotoren. Zum Anschießen der Servos befinden sich drei Stiftkontakte auf dem Shield. Die Widerstände R1, R2 und R3 schützen den Arduino, falls ein Servo versehentlich verpolt angeschlossen wird. R5, R6 und R7 sorgen für stabile Signale auf den Datenleitungen der

Inverse Kinematik



Eine kinematische Kette ist in der Robotik das mathematische Modell eines mechanischen Systems, das aus einer Gesamtheit starrer Körper und Gelenke besteht. Die Bewegungen einer solchen kinematischen Kette können anhand mathematischer Modelle modelliert werden.
Direkte Kinematik bedeutet, dass die Konfiguration der kinematischen Kette auf der Basis der Gelenkpositionen berechnet wird. Inverse Kinematik ist das genaue Gegenstück, hier werden mit einer gewünschten kinematischen Kette als Ausgangsgröße die Positionen der Gelenke berechnet. Das Ziel ist meistens, die Position und den Weg des Endeffektors oder Manipulators zu bestimmen.
Außerhalb der Robotik wird die inverse Kinematik auch bei 3D-Animationen und Computerspielen angewendet. Bewegungen menschlicher und tierischer Körper können ebenfalls auf solche Weise modelliert werden. Wenn wir diese Theorie auf die Sanduhr anwenden, besteht die kinematische Kette aus vier starren Körpern und fünf Scharniergelenken, wie in der Skizze dargestellt. Zwei der fünf Scharniergelenke besitzen einen Aktuator (= Servomotor). Der Endeffektor ist in diesem Fall der Schreibstift. Inverse Kinematik bedeutet hier, dass anhand einer gewünschten Position des Schreibstifts (x_6,y_6) die Winkel der Aktuatoren θ_1 und θ_2 (= Stellungen der Servomotoren) berechnet werden. Da die Scharniergelenke nur einen Freiheitsgrad besitzen und nur zwei Aktuatoren existieren, ist dies durch mathematische Modelle mit nur wenig Aufwand erfassbar, so dass ein ATmega328P auf einem Arduino-Board diese Aufgabe übernehmen kann.
Um die Berechnungen zu vereinfachen, haben der linke und rechte Servomotor die gleiche Y-Position $y_1 = y_2$.
Die Grafik stellt unterschiedliche Positionen, Längen und Winkel dar. Einige Größen sind konstant, andere sind variabel.

Konstant: (x_1,y_1), (x_2,y_2), L_{13}, L_{24}, L_{35}, L_{45}, L_{46}, L_{56}, α_{564}
Variabel: (x_5,y_5), (x_6,y_6), L_{15}, L_{26}, α_1, β_1, θ_1, α_2, β_2, θ_2, α_5, α_6

Zuerst bestimmen wir den Winkel θ_2. Diesen Winkel leiten wir ab von α_2 und β_2. α_2 ist der Winkel zwischen der X-Achse und der Geraden, die durch P_2 (rechter Servomotor) und P_6 (Stift) verläuft:

$$\alpha_2 = \arctan 2\left(\frac{y_6 - y_2}{x_6 - x_2}\right)$$

Winkel β_2 berechnen wir anhand der Seiten des Dreiecks P_2-P_4-P_6. Länge L_{46} ist bekannt. Länge L_{26} berechnen wir wie folgt:

$$L_{26} = \sqrt{(x_6 - x_2)^2 + (y_6 - y_2)^2}$$

Nun können wir Winkel β_2 bestimmen:

$$\beta_2 = \arccos\left(\frac{L_{26}^2 + L_{24}^2 - L_{46}^2}{2 \cdot L_{26} \cdot L_{24}}\right)$$

Winkel θ_2 ist dann:

$$\Theta_2 = \alpha_2 - \beta_2$$

Winkel θ_1 berechnen wir anhand des Dreiecks P_1-P_3-P_5. Die Position von P_5 ist (x_5,y_5), sie ist eine Variable, die wir erst berechnen müssen.
Dreieck P_4-P_5-P_6 ist ein starrer Körper, Winkel und Seiten sind konstant. Um (x_5,y_5) zu berechnen, gehen wir davon aus, dass das Dreieck P_4-P_5-P_6 mit der Seite L_{56} um P_6 rotiert über einen Winkel α_5, wie in der Grafik angegeben. Zuerst bestimmen wir Winkel α_6:

$$\alpha_6 = \arccos\left(\frac{L_{26}^2 + L_{46}^2 - L_{24}^2}{2 \cdot L_{26} \cdot L_{46}}\right)$$

Winkel α_2 war bereits bekannt. Nun können wir Winkel α_5 bestimmen:

$$\alpha_5 = \pi + \alpha_2 + \alpha_6 - \alpha_{564}$$

Die Position von P_5 ist dann wie folgt:

$$x_5 = x_6 + L_{56} \cdot \cos(\alpha_5)$$
$$y_5 = y_6 + L_{56} \cdot \sin(\alpha_5)$$

Zurück zu Winkel θ_1. Diesen Winkel leiten wir von α_1 und β_1 ab. Winkel α_1 ist der Winkel zwischen der X-Achse und der Geraden, die durch P_1 (linker Servomotor) und P_5 verläuft:

$$\alpha_1 = \arctan 2\left(\frac{y_5 - y_1}{x_5 - x_1}\right)$$

Winkel β_1 berechnen wir anhand der Seiten des Dreiecks P_1-P_3-P_5. Länge L_{35} ist bekannt. Länge L_{15} berechnen wir wie folgt:

$$L_{15} = \sqrt{(x_5 - x_1)^2 + (y_5 - y_1)^2}$$

Winkel β_1 ist dann:

$$\beta_1 = \arccos\left(\frac{L_{15}^2 + L_{13}^2 - L_{35}^2}{2 \cdot L_{15} \cdot L_{13}}\right)$$

Schließlich folgt θ_1:

$$\Theta_1 = \pi - \beta_1 - \alpha_1$$

Diese Berechnungen sind in der Funktion **pen_calc** des Sketches implementiert.

Servomotoren in den Zeiten, in denen sich der Arduino-Mikrocontroller im Reset-Zustand befindet.
Die aktuelle Zeit läuft mit einem Echtzeit-Uhrenchip weiter, beispielsweise wenn der Netzstecker kurz gezogen und wieder eingesteckt wird. Die Sanduhr muss dann nicht wieder neu gestellt werden. Der Uhrenchip ist ein PCF2129A, der nicht nur wenig kostet, sondern wegen der internen Temperaturkompensation auch eine hohe Ganggenauigkeit hat. Die Abweichung beträgt laut Datenblatt nur 3 ppm. Eine Knopfzelle CR2032 hat als Backup-Batterie eine Lebensdauer von etwa zehn Jahren.
Die Betriebsspannung des Uhrenchips beträgt 3,3 V, sie wird vom Arduino bereitgestellt. Die I²C-Anschlüsse sind für den Signalpegel 5 V tolerant, so dass sie problemlos mit der I²C-Schnittstelle des Arduino verbunden werden dürfen. Eine Pegelanpassung ist nicht erforderlich.

Arduino-Sketch

Der Sketch der Sanduhr wurde für den Arduino Uno geschrieben, er läuft auch auf dem Elektor Uno R4. Im Sketch ist die Funktionalität implementiert, die zum Kalibrieren der Sanduhr nach dem Aufbau nötig ist. Der Aufbau wird ausführlich in der separaten Bauanleitung beschrieben, die ebenso wie der Sketch von [2] heruntergeladen werden kann. Außerdem enthält der Sketch natürlich die Uhrenfunktion, die die Zeit in den Sand setzt. Auf eine Reihe interner Sketch-Funktionen ist der Zugriff über Kommandos möglich, so dass der Schreibstift beliebig bewegt werden kann, beispielsweise um Figuren in den Sand zu malen oder die Rüttelmotoren zu aktivieren. Im Modus *Autonom* läuft die Sanduhr selbstständig, gesteuert vom Echtzeituhr-Chip. Der Modus *Kommando* bedeutet, dass die Sanduhr auf Kommandos reagiert.
Nachdem der Sketch in den Arduino geladen ist, können mit dem Monitor der Arduino IDE Kommandos zur Sanduhr gesendet werden. Die Übertragungsgeschwindigkeit muss auf 9600 Baud eingestellt werden, und *No line ending* (kein Zeilenende) darf nicht angehakt sein. Die Kommandos bestehen aus kurzen Strings (beispielsweise **svd**), gegebenenfalls gefolgt von einem oder mehreren Parametern (wie **ps -20.55 +55.8**). Einen Bogen, um ein anderes Beispiel zu nennen, schreibt das Kommando **pa** in den Sand. Mit Drücken der Enter-Taste wird ein Kommando gesendet. Bei den meisten Kommandos gibt der Arduino eine Antwort zurück. Dokumentiert sind die Kommandos im ersten Teil des Sketch-Textes.

Noch einmal zurück zum Anfang. Während des Aufbaus muss eine Kalibrierung vorgenommen werden. Zur Prozedur gehört die Eingabe einer Reihe von Kommandos, gegebenenfalls begleitet von Parametern. Nachdem die Einstellungen durchgeführt sind (Kommando **sed**), werden sie im EEPROM des Mikrocontrollers gespeichert (Kommando **sew**). Beim Starten prüft der Sketch, ob der EEPROM-Inhalt gültig ist. Trifft dies zu, hängt von den Einstellwerten ab, in welchem Modus die Uhr startet. Vermutlich ist beim ersten Start der Modus *Autonom* eingestellt. Bei ungültigem EEPROM-Inhalt startet die Uhr im Modus *Kommando*, ebenso wie beim ersten Start nach dem Laden des Sketches. Wird das EEPROM beim späteren, erneuten Laden des Sketches nicht gelöscht,

Stückliste Sanduhr

Widerstände:
R1...R3 = 220 Ω, SMD 0805
R4 = 1k8, SMD 0805
R5...R9 = 4k7, SMD 0805
R10 = 100 k, SMD 0805

Kondensatoren:
C1...C4 = 100 n, SMD 0805 MLCC
C5 = 10 µ/10 V, SMD 1206 MLCC
C6 = 22 µ/16 V

Halbleiter:
D1...D3 = S2J-E3
T1 = BCX56
IC1 = DC/DC-Wandler SIP3 5 V/1 A (Würth Elektronik 173 010 578)
IC2 = PCF2129A

Außerdem:
K1 = Satz Stiftkontaktleisten, gerade (1 · 10-polig, 2 · 8-polig, 1 · 6-polig), RM 2,54 mm
SERVO_Z,SERVO_L,SERVO_R = Stiftkontaktleiste abgewinkelt, 3-polig, RM 2,54 mm
B1 = Knopfzelle CR2032 mit Halter
Arduino Uno R3 oder vergleichbar

Mechanische Bauteile:
6 · Schraube M2x10, Phillips-Pozidrive
6 · Schraube M2,5x8 verzinkt Stahl, Pozidrive DIN 7985A
6 · Schraube M2,5x12 verzinkt Stahl, Pozidrive DIN 7985A
2 · Schraube M3x6 verzinkt Stahl, Pozidrive DIN 7985A
2 · Schraube M3x8 verzinkt Stahl, Pozidrive DIN 7985A
15 · Schraube M3x10 verzinkt Stahl, Pozidrive DIN 7985A
1 · Schraube M4x30 Kunststoff, Phillips (Ende anspitzen mit Bleistiftspitzer)
6 · Sechskantmutter M2 verzinkt Stahl, DIN 934
7 · Sechskantmutter M3 Stahl, DIN 934
3 · Gegenmutter M3 verzinkt Stahl, DIN 985
1 · Sechskantmutter M4 Polyamid
2 · Unterlegscheibe M3 verzinkt Stahl, DIN 125A
4 · Unterlegscheibe M3 Kunststoff, DIN 125A
4 · Abstandsrolle 3 mm hoch, Polyamid, für M3
4 · Abstandshalter, Sechskant, 25 mm M/F M3 Messing vernickelt (min. 33 mm hoch, z. B. TME TFM-M3X25/DR213)
2 · Kabelschelle Polyamid, Panduit CCS25-S10-C
4 · Kunststofffuß, selbstklebend
3 · Micro-Servo, Tower Pro MG90S oder MG90 mit Metall-Zahnrädern
2 · Rüttelmotor rund, Durchmesser 6 mm, VM6ZK273 (450-007 JPR Electronics)
Extrudiertes (XT) PMMA, 3 mm stark, lasergeschnitten
Feiner weißer Sand (Silbersand)
Lebensmittelfarbe (optional)
oder
Bauteilesatz mit sämtlichen Teilen einschließlich Shield-Platine (SMDs vormontiert), erhältlich im Elektor-Shop

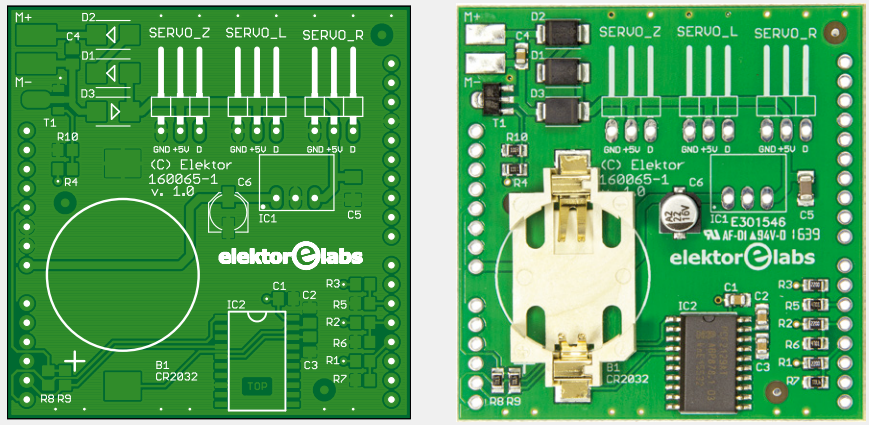


Bild 4. Die zugehörige Platine ist überwiegend mit SMDs bestückt, auf der Platine des Bauteilesatzes sind sie bereits vormontiert.

entfällt das Eingeben und Speichern der Einstellungen mit dem Kommando **sew**. Falls doch einmal nötig, lässt sich der EEPROM-Inhalt mit dem Kommando **sec** löschen.

Im Modus *Autonom* setzt die Uhr periodisch die aktuelle Zeit in den Sand. Die Prozedur besteht aus einer Folge horizontaler und vertikaler Bewegungen des Zeichenstifts. Die aktuelle Stiftposition in der X- und Y-Ebene dient dem Sketch als Startpunkt der Bewegungen: Eine Linie zeichnen, einem Bogen folgen, ein Symbol zeichnen oder den Stift räumlich verschieben.

Die Funktion **pen_set** bewegt den Stift zur angegebenen XY-Position. Zuerst werden die Koordinaten abhängig vom Stand des linken und rechten Servomotors in Radianten umgerechnet. Die Berechnung, ausgeführt von der Funktion **pen_calc**, kann erfolgreich sein oder fehlschlagen. Fehl schlägt die Berechnung, wenn die angegebenen Koordinaten mit dem Stift nicht erreichbar sind. Bei erfolgreicher Berechnung werden die resultierenden Motorstellungen von Radianten in PWM-Impulsbreiten für den linken und rechten Servomotor umgerechnet. Zum Berechnen der inversen Kinematik wendet der Sketch Fließkomma-Prozeduren an. Sie sind im AVR-GCC-Compiler festgeschrieben, der den Sketch kompiliert. Die Maßeinheit für sämtliche Positionen und Abstände ist der Millimeter, die Maße stimmen mit den realen Maßen überein. Der Sketch enthält Definitionen physischer Abmessungen der mechanischen Elemente der Uhr. Winkel werden in Radianten ausgedrückt.

Die Zeichnung in **Bild 5** gibt die Abmessungen des Sandbetts und die Positionen des linken und rechen Servomotors in Millimetern an, wie sie im Sketch definiert sind.

Beim Schreiben in den Sand wird der Stift in Schritten bewegt, die maximal 0,25 mm weit sind. Die umfangreiche Rechenarbeit erledigt der Mikroprozessor in kürzester Zeit. Deshalb wird zwischen jede Positionsänderung eine Pause von 5 ms eingeschoben, anderenfalls würde sich der Stift zu schnell durch den Sand bewegen. Der Modus *Autonom* lässt sich jederzeit abbrechen, indem das Zeichen *Zeilenende* gesendet wird. Das Drücken der Taste *Enter* im Monitor bewirkt, dass der Sketch in den Modus *Kommando* wechselt. Der Übergang in diesen Modus kann verzögert sein, solange eine Schreibaktion in den Sand stattfindet. Mit dem Kommando **ma** kehrt der Sketch zum Modus *Autonom* zurück.

Aufbau

Im Rahmen dieses Beitrags können wir auf den Bau der Sanduhr nur beschränkt eingehen. Falls Sie die Herausforderung annehmen wollen und den Bau wagen, verweisen wir auf die ausführliche und mit vielen Fotos illustrierte Bauanleitung [2]. Zuerst montieren Sie die Kontaktleisten und den DC/DC-Wandler auf das Arduino-Shield. Stecken Sie das Shield auf den Arduino und schließen Sie die Servomotoren an. Achten Sie auf die Positionen der Steckverbinder. Verbinden Sie den Arduino mit dem Steckernetzteil und über USB mit dem PC, auf dem die Arduino IDE installiert ist. Kompilieren Sie den Sketch [2] und laden Sie ihn in den Arduino. Nachdem das geschehen ist und der Arduino neu gestartet wurde, werden die Servomotoren selbsttätig in die Mittelstellung gefahren (Impulsbreite 1500 µs). Unterbrechen Sie die Verbindung zum PC und versetzen Sie den Arduino in den stromlosen Zustand. Entfernen Sie das Shield und koppeln Sie die Servomotoren ab. Wenn Sie über einen Servotester verfügen, können Sie die Servomotoren auch mit dem Servotester in Mittelstellung bringen.

Nun ist es an der Zeit, die Mechanik aufzubauen. Kleben Sie die vier Kunststofffüße unter die Grundplatte und montieren Sie die vier Stützen für das Sandbett. Montieren Sie anschließend den Lift-Servomotor an den Grundrahmen, anschließend montieren Sie die Konstruktion zusammen mit dem Arduino und dem Shield auf die Grundplatte.

Jetzt sind der linke und rechte Servomotor des Pantograf-Mechanismus an der Reihe. Schließen Sie die Servomotoren wieder am Shield an und montieren Sie die Arme und den Schreibstift des Pantografen. Bei der Montage der Arme werden die Schrauben unmittelbar in den PMMA-Kunststoff gedreht. Wegen der schon erwähnten thermoplastischen Eigenschaften lässt sich dies leicht bewerkstelligen, denn der Kunststoff passt sich durch die Reibungswärme beim Einschrauben dem Schraubengewinde an. Diese Montagemethode hat gleichzeitig den Vorteil, dass das mechanische Spiel minimiert wird. Die für den Schreibstift vorgesehene Montagebohrung ist dagegen mit einem Gewinde versehen. Wenn Sie die mechanischen Teile in Eigenregie mit einem Lasercutter schneiden, müssen Sie in diese Bohrung ein M4-Schraubgewinde eindrehen.

Montieren Sie das Sandbett, lassen Sie jedoch vorläufig die Rüttelmotoren beiseite, denn zuvor steht die Kalibrierung auf dem Arbeitsplan. Verbinden Sie das Steckernetzteil mit dem Arduino und stecken Sie es in die Netzsteckdose. Verbinden Sie den Arduino über USB mit dem PC. Über den Monitor der Arduino IDE können Sie jetzt Kommandos eingeben, um die Servomotoren zu positionieren. Um die Pantograf-Servomotoren zu kalibrieren, bestimmen Sie die Positionen, in denen die Servomotoren exakt waagerecht und senkrecht stehen. Diese Positionen werden im EEPROM des Mikrocontrollers gespeichert. Anschließend wird der Lift-Servomotor kalibriert. Platzieren Sie das Sandbett und positionieren Sie

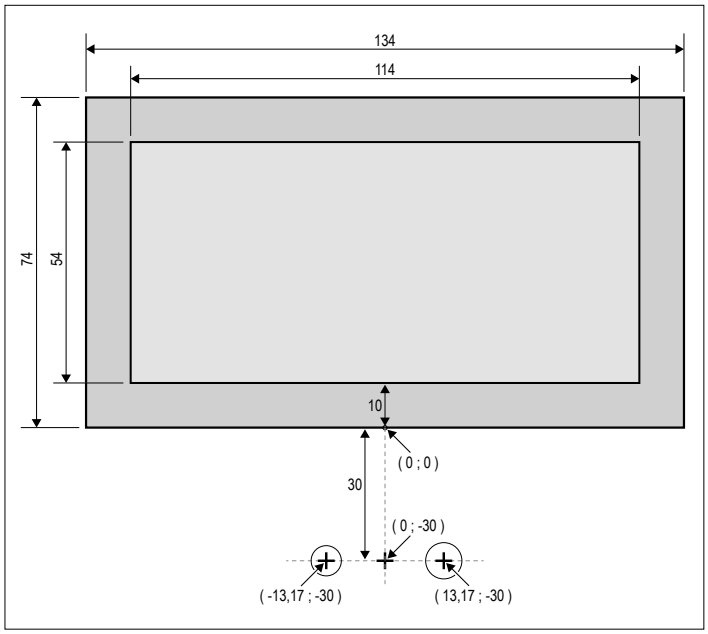


Bild 5. Die Positionen des linken und rechten Servomotors in Millimetern, bezogen auf das Sandbett, wie sie im Sketch definiert sind.

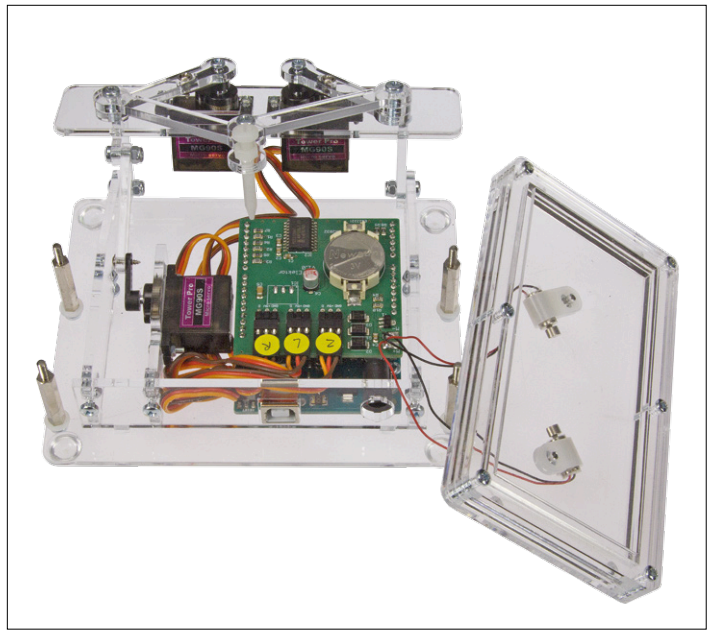


Bild 6. Dieses Foto illustriert die mechanische Konstruktion der Sanduhr.

den Lift-Servomotor so, dass der Schreibstift wenige Millimeter über dem Boden schwebt, ungeachtet der XY-Position. Sie können später eine Feinjustierung vornehmen, indem Sie den Schreibstift höher oder niedriger einspannen. Drehen Sie die M4-Mutter fest, wenn sich der Schreibstift in der gewünschten Position befindet. Schließlich werden noch die mittlere und höchste Position des Lift-Servomotors eingestellt. Diese Einstellung ist nicht kritisch, Sie können auf die Werte aus der separaten Bauanleitung zurückgreifen.

Speichern Sie die Einstellungen im EEPROM und stellen Sie anschließend die korrekte Zeit ein. Beim ersten Test kann die Sanduhr vorläufig ohne Sand laufen. Wenn der Test erfolgreich war, stellen Sie den Modus *Autonom* ein, so dass die Sanduhr in festen Intervallen selbsttätig die Zeit schreibt. Die USB-Verbindung zum PC können Sie entfernen.

Nehmen Sie noch einmal die Betriebsspannung vom Arduino und montieren Sie die Rüttelmotoren unter dem Sandbett. Schließen Sie die Zuleitungen der Motoren parallel zueinander an und löten Sie die Leitungen an die Lötinseln auf dem Shield. Achten Sie darauf, dass Sie Teile aus PMMA-Kunststoff nicht versehentlich mit dem heißen Lötkolben berühren. Platzieren Sie dann das Sandbett auf die Stützen und stellen Sie sicher, dass die Schwingmotoren frei drehen können und die Anschlussleitungen nicht berühren.

Als krönenden Abschluss gießen Sie den Sand in das Sandbett: Endlich ist es geschafft!

Falls Sie Freude daran haben, können Sie vorher noch den Sand einfärben. Schütten Sie den Sand in einen verschließbaren Behälter aus Glas und geben Sie einige Tropfen Lebensmittelfarbe hinzu. Schütteln Sie den Behälter kräftig und so lange, bis der Sand die Farbe gleichmäßig angenommen hat. Die Prozedur können Sie nach Hinzugeben einer weiteren Farbe wiederholen, solange bis sich der Sand in Ihrem Lieblingsfarbton zeigt. Gießen Sie den Sand erst in das Sandbett, nachdem er vollständig durchgetrocknet ist. ⏮

(160065)gd

Weblinks

[1] www.elektormagazine.de/160065

[2] www.elektor.de/sandclock-160065-71

Anzeige

HAMMOND MANUFACTURING®

Gehäuse und Platform für Pi und Arduino

www.hammondmfg.com/1593HAM.htm

www.hammondmfg.com/1593HAMEGG.htm

01256 812812

sales@hammondmfg.eu

Internetradio mit Raspberry Pi

Tune in und chill out

Von **Roy Aarts** (Elektor-Labor)

Mit diesem Selbstbauprojekt kann man jedem Online-Radiosender lauschen.

Eigenschaften

- Wiedergabe von Audio-Streams von Internet-Radiosendern
- Verwendet den vorhandenen Audio-Ausgang
- Funktioniert PC-unabhängig
- Erstellung einer eigenen Senderliste
- Selbstbau ohne Löten

Internetradios gibt es heute in allen denkbaren Formen, Farben und Varianten. Einige davon sind richtig teuer. Mit diesem Projekt kann man sich aber selbst solch einen Empfänger bauen – sogar recht preiswert auf der Basis eines Raspberry-Pi-2-Boards. Aus seinem Kopfhörer-Anschluss kommen dann die dekodierten Audiosignale von Radio-Streams aus dem Internet. Neben einem RPi benötigt man einen Touch-Screen mit 2,8" Diagonale, eine SD-Karte und optional noch einen WLAN-Dongle. Die Bedienoberfläche wurde in Python mit Hilfe von *pygame* erstellt. Der Code ist gut nachvollziehbar und kann an eigene Bedürfnisse angepasst werden.

Installation

Zunächst muss man *Raspbian* auf dem RPi installieren. Nach dem Download von *Raspbian Jessie* [1] via PC schreibt man mit *Win32DiskImager* [2] ein Jessie-Image auf eine leere SD-Karte (minimale Kapazität: 4 GB). Anschließend steckt man die SD-Karte in den Karten-Slot des RPi, schließt einen Monitor, Maus, Tastatur und WLAN-Dongle (oder Ethernet per Kabel) an und schaltet das System ein. Nach dem RPi-Bootvorgang stellt man die Netzwerkverbindung ein (Klick auf das Icon in der oberen rechten Ecke von **Bild 1**). Steht die Verbindung, notiert man sich die IP-Adresse des RPi, denn diese Info kann später noch nützlich sein. Hierzu bewegt man den Mauszeiger über das Icon mit der Netzwerkverbindung.
Jetzt werden die Libraries für den Touchscreen installiert. Falls dieser noch nicht angeschlossen sein sollte, holt man dies nach – aber hierzu sollte man den RPi stromlos gemacht haben (Herunterfahren: linke obere Ecke und *Shutdown* auswählen)! Nach Anschluss des Touchscreens anstelle des Monitors schaltet man den RPi wieder ein. Nun werden die erforderlichen Treiber via PC installiert. Nach dem Download und Start von *PuTTY* [3] gibt man die IP-Adresse des RPi bei „Host address (or IP address)" ein. Wenn eine Benachrichtigung aufpoppt, klickt man auf „Yes". Nun sollte sich ein Terminal-Fenster zum Einloggen mit dem Username „pi" und dem Password „raspberry" öffnen. Jetzt gibt man ein:

```
sudo wget -N https://github.com/
   watterott/RPi-Display/raw/master/rpi-display.sh
```

und betätigt Enter. Dadurch erfolgt der Download der nötigen Dateien von Github. Nach dem Erscheinen des Prompts tippt man:

```
sudo /bin/bash rpi-display.sh 270
```

und die Github-Dateien werden installiert.

Nach einiger Zeit erscheinen Fragen. Man beantwortet sie mit:

- Activate the console on the TFT display? y/n **Y**
- Install fbcp (Framebuffer Copy)? y/n **N**
- Install xinput-calibrator? y/n **Y**
- Install tslib (Touchscreen library)? y/n **Y**
- Calibrate Touchscreen now? y/n **N**
- Reboot the system now? y/n **Y**

RPi bootet dann und die Verbindung mit dem Computer geht verloren. Wenn alles geklappt hat, erscheint nach dem Booten von RPi die Bedienoberfläche auf dem Touchscreen.

Kalibrierung & Music Player

Um den Touchscreen zu konfigurieren verbindet man sich über PuTTY vom PC aus mit dem RPi, loggt sich ein und tippt:

```
sudo TSLIB_FBDEVICE=/dev/fb1 TSLIB_TSDEVICE=/dev/
   Eingang/Touchscreen ts_calibrate
```

Auf dem Touchscreen erscheint das Kalibrationsprogramm (**Bild 2**). Nach diesen Schritten ist der Touchscreen kalibriert.

Jetzt wird der Music-Player-Client *mpc* und der Daemon installiert. Hierzu dienen folgende Befehle:

```
sudo apt-get update
sudo apt-get install mpd mpc
```

User-Interface

Nun muss noch die Bedienoberfläche installiert werden. Die Wahl fällt auf *winSCP* [4]; der Download ist auch über die Elektor-Webseite zu diesem Projekt [5] verfügbar. Wir müssen nun die Dateien auf die SD-Karte kopieren. Dazu beendet man PuTTY und startet winSCP. Dort loggt man sich wie zuvor bei PuTTY ein. Bei einer Benachrichtigung klickt man auf „Yes". Anschließend kopiert man den Ordner „pi-radio" des Downloads auf den RPi (zum Beispiel in das Verzeichnis /home). Dann kopiert man die Dateien „launcher.sh" und „shutdown.sh" in das Desktop-Verzeichnis des RPi. Diese beiden Dateien dienen dazu, das Radio zu starten und den RPi herunterzufahren. Jetzt schließt man winSCP und startet PuTTY.

Die beiden auf den RPi-Desktop kopierten Dateien haben noch keine anklickbaren Icons. Dies wird mit PuTTY so gefixt:

```
sudo chmod +x /home/pi/Desktop/launcher.sh
```

und

```
sudo chmod +x /home/pi/Desktop/shutDown.sh
```

Sender

Das Internetradio ist jetzt fertig installiert, aber noch kann man keinen Sender hören. Hierzu benötigt man den Link zu dessen Audio-Stream. Eine gute Webseite mit einer Liste solcher Links ist *SomaFM* [6]. Hier finden sich viele freie Radios unterschiedlicher Genres. Um via SomaFM einen brauchbaren Link zu erhalten, klickt man auf den gewünschten Sender und dann auf „Direct stream links". Der benötigte Link-Typ ist „Direct server".
Man fügt einen Sender hinzu, indem man PuTTY startet und den Befehl `mpc add` gefolgt vom URL des Senders eingibt. Ein Beispiel:

```
mpc add http://ice1.somafm.com/bagel-128-mp3
```

Links mit der Endung „.m3u" funktionieren nicht. Im Netz finden sich viele andere Radio-Listen; man ist also nicht auf SomaFM beschränkt. Die URLs lassen sich sehr einfach testen, denn man muss sie nur in den Browser auf dem PC eingeben. Bei einem gültigen Link öffnet sich ein Player, und schon hört man diesen Sender.

Zur Auflistung der schon gespeicherten Sender dient das Kommando `mpc playlist`. Um einen Eintrag aus der Liste zu löschen, tippt man `mpc del` gefolgt von der Nummer der Liste in PuTTY ein. Ein Beispiel: `mpc del 2` entfernt die zweite Position der Liste.

Bedienung

Jetzt kann der Radio-Player durch Doppelklick auf sein Icon auf dem Touchscreen und einen Druck auf „Execute" gestartet werden. Das Interface (**Bild 3**) ist zwecks Fingerbedienung sehr einfach und benötigt daher keinen Eingabestift (**Bild 2**). Verfügbar sind die Befehle:

- *Play* – startet mpc mit der aktuellen Senderliste.
- *Pause* – stoppt mpc.
- *Volume down* – vermindert die Lautstärke.
- *Volume up* – erhöht die Lautstärke.
- *Mute* – setzt die die Lautstärke auf null.
- *Refresh* – stoppt und startet den Player neu, erneuert die Darstellung.
- *Exit* – beendet das Player-Interface, aber mpc läuft weiter.

Zum bequemen Herunterfahren genügt ein Doppelklick auf das Icon „shutDown" und die Betätigung mit Execute. Viel Spaß beim Radiohören!

(160043)

Weblinks

[1] www.raspberrypi.org/downloads/raspbian

[2] http://sourceforge.net/projects/win32diskimager

[3] www.chiark.greenend.org.uk/~sgtatham/putty/download.html

[4] http://winscp.net/eng/download.php

[5] www.elektormagazine.de/160043

[6] http://somafm.com

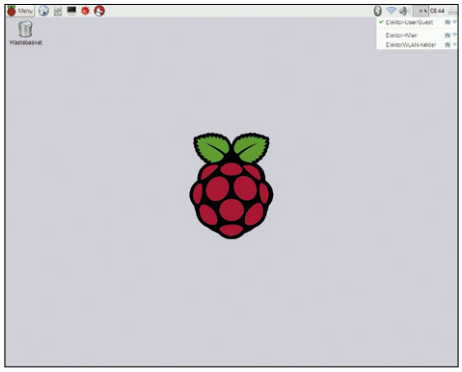


Bild 1. Hier konfiguriert man die RPi-Netzwerkeinstellungen.

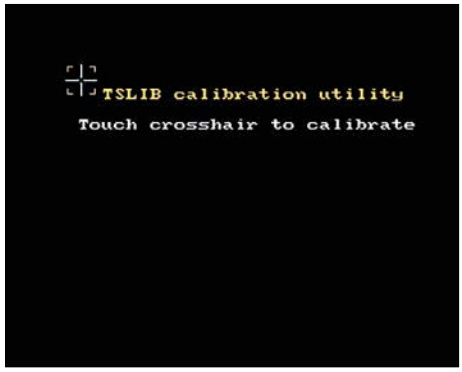


Bild 2. Vor dem Betrieb muss der Touchscreen kalibriert werden.

Bild 3. Internet-Radio mit Fingerbedienung — ein Eingabestift ist nicht nötig.

Schwebende Lampen (und andere Objekte)

Eine Experimentier-Anleitung

Von **Martin Oßmann** (D)

Wie kann man etwas scheinbar mühelos in der Luft schweben lassen? Da denkt der Elektroniker natürlich gleich an eine Anordnung mit (Elektro-)Magneten, die sich gegenseitig abstoßen. Leider ist es nicht so einfach, eine Schaltung zu entwerfen, mit der ein dauerhaftes, stabiles Schweben möglich ist. Hier werden einige Experimente und Schaltungen beschrieben, um so etwas zu realisieren. Gezeigt wird sogar eine integrierte induktive Energie-Übertragung, um am schwebenden Objekt LEDs aufleuchten zu lassen.

Vor einiger Zeit entdeckte der Autor die Produktbesprechung einer „schwebenden Lampe“ (Flyte, siehe **Bild 1** [1]). Da war natürlich die Neugierde geweckt und es wurden zwei Exemplare bestellt, eins zum Vorführen und eins fürs „Reverse Engineering“.
Bevor die Lampen eintrafen, wurde schon ein wenig Zeit investiert, um zu erkunden, wie diese wohl funktionieren [2]. Das entscheidende Stichwort für eine Google-Anfrage ist „repulsive magnetic levitation“. Es taucht dann eine Reihe von Bildern und Videos auf, genaue Anleitungen hat der Autor aber nicht gefunden.

Daher dieser Artikel: Wir wollen im Detail beschreiben, wie das Ganze funktioniert. Dabei sollen alle notwendigen Hin-

Bild 1. Die Schwebende Lampe namens Flyte.

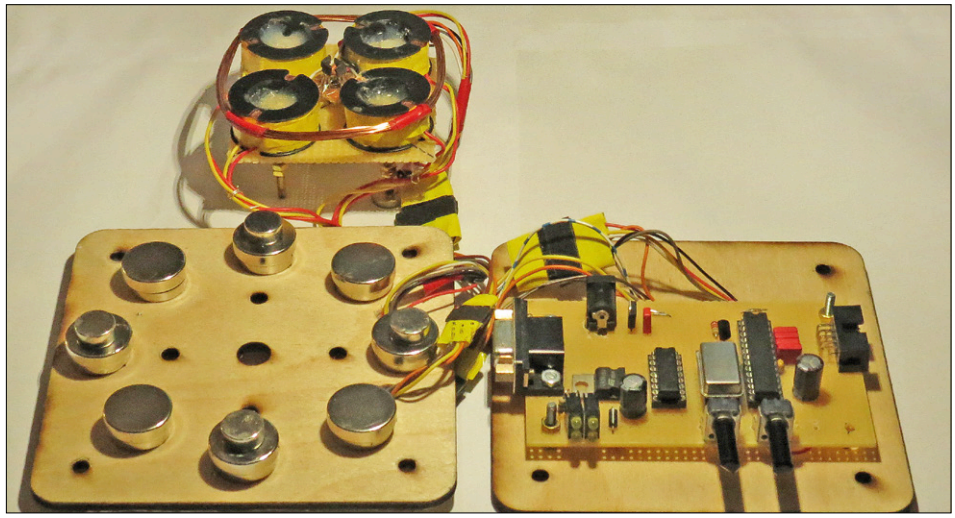


Bild 2. Bestandteile der fliegenden Untertasse.

weise gegeben werden, damit der Leser so etwas selbst nachbauen kann. Außerdem wollen wir uns die Flyte-Lampe genauer ansehen.

Wie funktioniert's?

Allen Aufbauten gemeinsam ist die Verwendung eines Ringmagneten, um die abstoßende Kraft auf einen schwebenden Magneten zu erzeugen. Beschafft man sich einen solchen Ringmagneten und hält zum Beispiel eine Magnetkugel in den Bereich oberhalb der Mitte, so kann man unmittelbar die abstoßende Kraft spüren. Allerdings versucht der Schwebemagnet, radial auszuweichen.

Um das zu verhindern, wird eine elektronische Regelung eingesetzt, um das Schwebeobjekt in der x-y-Ebene zu stabilisieren. Es kommen meist vier Elektromagnete zum Einsatz, zwei für die x-Richtung und zwei für die y-Richtung. Diese sind paarweise antiparallel oder antiseriell geschaltet. Damit kann man dann Kräfte in x- und y-Richtung auf den Schwebemagnet ausüben. Die Position des Schwebemagneten wird dabei meistens von zwei oder vier Hallsensoren erfasst.

Laut einer Quelle reicht durchaus ein Arduino zur Regelung aus, so dass der Verfasser sein Glück auch mit einem Atmel AVR versucht hat. Der Einsatz einer Regelung ist dabei eine der Möglichkeiten, dem Theorem von Earnshaw [3] zu entgehen, welches besagt, dass alleine mit Dauermagneten kein schwebender Magnet realisiert werden kann. In **Bild 2** sieht man die typischen Bestandteile, eine Platte mit einem Ring aus Dauermagneten, eine Platte mit Elektromagneten, sowie die Regelelektronik. In **Bild 3** ist der Aufbau schematisch dargestellt.

Es hat sich bewährt, die Dauermagnete und die Elektromagnete auf getrennte Platten zu montieren (P1 und P2), deren Abstand d durch Schrauben justierbar ist. So kann man die Wirkung der beiden Magnetarten optimal kombinieren.

Erste Testaufbauten

Will man selbst Magnete schweben lassen, tut man gut daran, eine gehörige Sammlung verschiedenster Magnete zu beschaffen (zum Beispiel bei supermagnete.de). Dabei sollte man auch Verluste einkalkulieren. Schnappen nämlich zwei Neodym-Magnete mit voller Wucht aufeinander, kann einer schnell wegen der Sprödigkeit zu Bruch gehen.

Da man die Elektromagnete innerhalb des Ringmagneten platzieren sollte, braucht man einen Ringmagnet mit relativ großem Innendurchmesser, der schwer beschaffbar ist. Daher hat der Autor statt des Ringmagneten eine kreisförmige Anordnung von vier oder acht kreisförmigen Dauermagneten benutzt. Dabei werden Magnete mit einem Zentralloch an einer Platte festgeschraubt (Bild 2), und auf die Lochmagnete zur Verstärkung noch Scheibenmagnete gesetzt. Durch zusätzlich aufgesetzte Scheibenmagnete kann man das Magnetfeld auf einfache Weise feintunen.

Der Innendurchmesser des Rings aus Magneten liegt zwischen 80 und 100 mm. Damit kann man schon ganz beträchtliche abstoßende Kräfte erreichen. Mit dieser Magnetplatte ist die erste Zutat gebaut, die Abstoßung in z-Richtung ist gesichert.

Ein wenig Theorie

Manch einer hat sich vielleicht schon gewundert, warum ein Ringmagnet zum Einsatz kommt, um die abstoßende Kraft zu erzeugen, und nicht ein Scheibenmagnet. Dazu wollen wir ein wenig in die Theorie der Abstoßung und Anziehung bei Magneten eingehen. Das Magnetfeld eines kreisförmigen Magneten kann man leicht (zum Beispiel mit FEMM) ausrechnen, es ist in **Bild 4** dargestellt.

Die rot/grünen Rechtecke sind die Querschnitte des Ringmagneten. In der Mitte senkrecht steht die Rotationsachse. Der rot/grüne Kreis symbolisiert eine (schwebende) Magnetkugel. Das Problem ist nun, dass dieses Feldbild nicht sehr aussagekräftig für die Kräfte auf Testmagnete ist. Das Magnetfeld selbst gibt nämlich nicht die Kraft, sondern das Drehmoment auf einen Testdipol (= Magnet) an. Ohne weitere Momente dreht der Dipol sich so, wie das Magnetfeld angibt.

Für die Kraft auf einen Dipol ist der Gradient (die Richtung und Stärke der größten Änderung) zuständig, wobei die Lage des Testmagneten auch noch eine Rolle spielt. Bei einem Scheibenmagneten ist es nun so, dass der Testmagnet sich immer in eine Position dreht, in der er in Richtung der Scheibe angezogen wird. Bei einem Ringmagneten ist das anders. Es gibt einen Bereich, in dem sich der Testmagnet in eine Lage dreht, bei der er tatsächlich abgestoßen wird.

Das heißt die Lage im Raum (Rotation um die Achsen) ist bis auf die Rotation um die z-Achse automatisch stabil. Unsere Elektromagneten kümmern sich dann um die x-y-Position und der Ringmagnet mit der Schwerkraft zusammen um die z-Position. Um die z-Achse kann sich das schwebende Objekt drehen, da dieser Freiheitsgrad nicht festgelegt ist.

Das gerade Gesagte kann man mit einem experimentellen Aufbau (vgl. **Bild 5**) gut demonstrieren. Dazu benutzen wir einen Ringmagneten und eine Magnetkugel. Die Magnetkugel befindet sich, frei drehbar, in einem Rohr in der Mitte des Ringmagneten. Damit fixiert das Rohr die Kugel in der x-y-Richtung. In der z-Richtung sorgt die abstoßende Kraft des Ringmagneten dann tatsächlich für ein Schweben, wobei der Kugelmagnet sich in eine Position dreht, in der die Abstoßung maximal wird.

Interessant ist übrigens, was passiert, wenn man den Aufbau aus Bild 5 auf den Kopf dreht. Aufgrund der abstoßenden Kraft

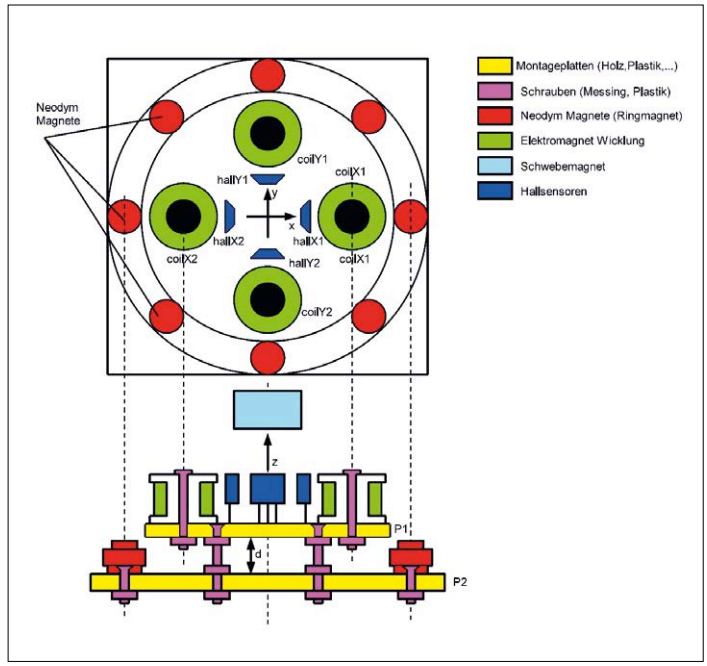


Bild 3. Schematischer Aufbau der mechanischen Konstruktion.

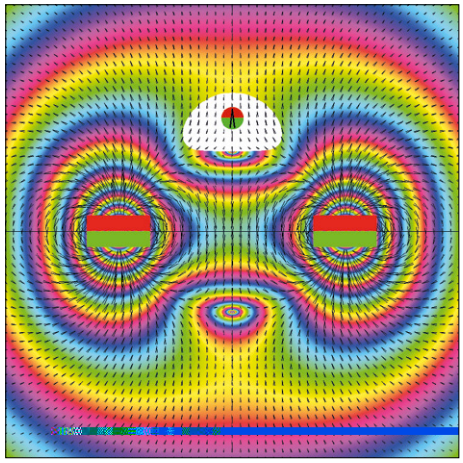


Bild 4. Magnetfeld des Ringmagneten.

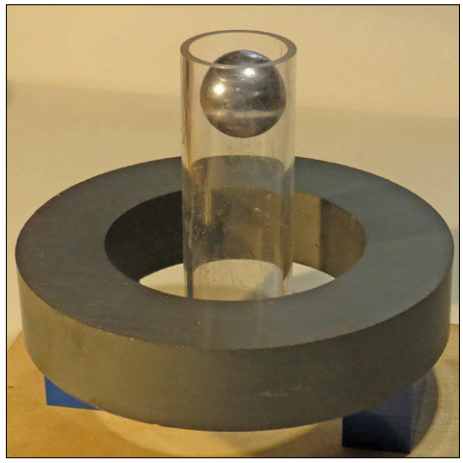


Bild 5. Der Kugelmagnet schwebt über dem Ringmagnet.

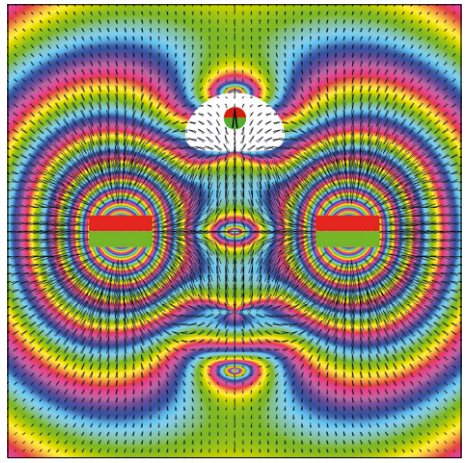


Bild 6. Kraftfeld des Ringmagneten.

würde man erwarten, dass die Kugel aus dem Plexiglasrohr heraus zu Boden fällt. Aber stattdessen entfernt sich die Kugel nur ein wenig vom Ringmagneten, und bleibt dann im Rohr, weil sie in diesem größeren Abstand schon vom Ringmagneten angezogen wird. Die Kugel befindet sich dann knapp oberhalb des weißen Bereiches aus Bild 4.

Ersetzt man den Ringmagneten durch einen Scheibenmagneten, knallt der Kugelmagnet immer mit voller Wucht auf den Scheibenmagneten, weil er sich im Raum so dreht, dass die Anziehung maximal wird. Die gerade plausibel gemachte Tatsache kann man auch mit einem „Pseudo-Feldbild" darstellen (**Bild 6**). Dabei ist die Kraft dargestellt, die sich ergibt, wenn der Testdipol sich in eine Lage gedreht hat, in der kein weiteres Drehmoment auf ihn wirkt. In Bild 6 kann man gut sehen, dass sich in der Mitte oberhalb des Ringes ein nicht allzu großer Abstoßungsbereich befindet (Weiß gekennzeichnet). In diesem kleinen Bereich wird sich später unser schwebender Magnet aufhalten.

Aufgrund dieser Zusammenhänge legt der Ringmagnet zusammen mit dem schwebenden Magnet die Schwebehöhe schon weitgehend fest. Hat man die Magnetplatte gebaut, kann man dann zum Beispiel mit einer Magnetkugel schon relativ gut erfühlen, wie hoch die Kugel später schweben wird. Die Größe des weißen Bereiches hängt dabei nicht von der Stärke des Ringmagneten ab, sondern von dessen Durchmesser. Um die Schwebehöhe zu vergrößern, muss man also einen größeren Ring von Magneten benutzen. Mit einem stärkeren Magneten kann man die Höhe nur innerhalb des weißen Bereiches vergrößern.

X-Y-Magnete

Als nächstes wollen wir auf die Elektromagnete zur x-y-Stabilisierung eingehen. Der Verfasser hat vier oder acht Elektromagnete verwendet, die mit 0,25 mm CuL-Draht gewickelt wurden. Als Spulenkörper kam entweder ein P36/22-Spulenkörper zum Einsatz (1200 Wdg), oder ein selbst 3D-gedruckter Spulenkörper aus Teilen nach Bild 7 und Bild 8 (1600 Windungen).

Diese Spulen sind dann zum Betrieb an 12 bis 15 V verwendbar. Die vier Spulen montiert man auf eine getrennte Platte, zum Beispiel aus Epoxy oder Lochrastermaterial.

Hallelemente

Um die Position des Schwebemagneten zu erfassen, kommen Hallelemente des Typs SS496A oder SS495A zum Einsatz. Man kann mit zwei oder vier Sensoren arbeiten, und man kann die Sensoren entweder waagerecht oder senkrecht in der Mitte montieren. Die besten Erfahrungen hat der Verfasser mit vier senkrecht montierten Hallsensoren gemacht, wobei die

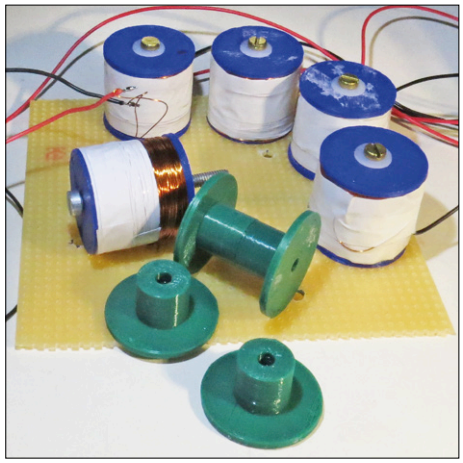


Bild 7. Wickelkörper aus dem 3D-Drucker.

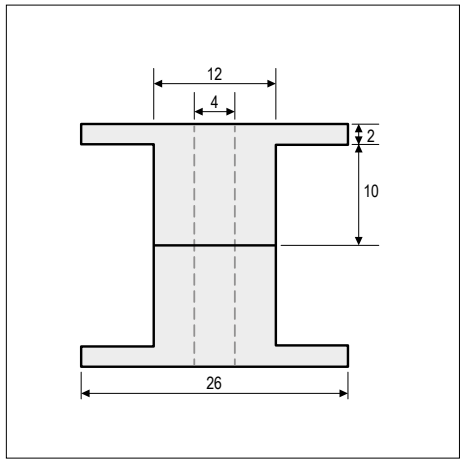


Bild 8. Maße der Wickelkörper.

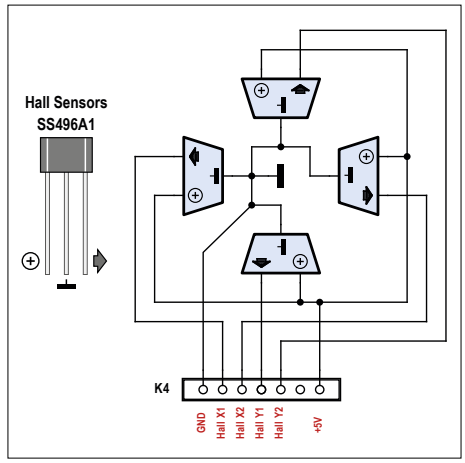


Bild 9. Verdrahtung der Hallsensoren.

Software je zwei gegenüberliegende Sensoren sozusagen in Brücke auswertet.
Die Sensoren montiert man an der Oberkante der Elektromagnete. Die Hallelemente werden dabei wie in **Bild 9** verschaltet und über den eingezeichneten Steckverbinder (6-polig) mit der Regelelektronik verbunden. Hat man die Elektromagnete mit den Sensoren aufgebaut, kann man mit der nachfolgenden Regelschaltung erste Tests machen.

Regelelektronik

Die gesamte Elektronik ist wie in **Bild 10** dargestellt aufgebaut. Ein ATmega Mikrocontroller erzeugt zwei PWM-Signalpaare, die den Motortreiber IC3 vom Typ L293D ansteuern. Dieser zweifache Vollbrückentreiber versorgt dann die zwei Spulenpaare mit Strom. Mit den antiparallel geschalteten LEDs kann man die aktuelle Stromrichtung in den Spulen kontrollieren.

Hält man den Schwebemagneten fest und bewegt ihn über die Hallelemente, dann ändert es sich, welche LED leuchtet, je nach Polung der Magneten.
Vier A/D-Eingänge werden zur Auswertung der Hall-Elemente benutzt, zwei A/D-Eingänge digitalisieren die Spannungen der zwei Potis R1 und R2, mit denen man die beiden Reglerkonstanten Kp (Proportionalteil) und Kd (Differentialteil) einstellen kann. Der Transistor T1 und der Kondensator C13 dienen der drahtlosen Energieübertragung. Ihre Funktion wird später genau erläutert.
In der Software laufen zwei unabhängige PD-Regler, je einer für die x- und die y-Richtung. Mit einer simplen RS232-Anpassschaltung kann man der Reglersoftware via RS232 bei 115.200 bit/s im Betrieb über die Schulter schauen. Es werden die folgenden Werte ausgegeben: Reglerkonstanten Kp und Kd, Position x und y, Stellgröße x und y.

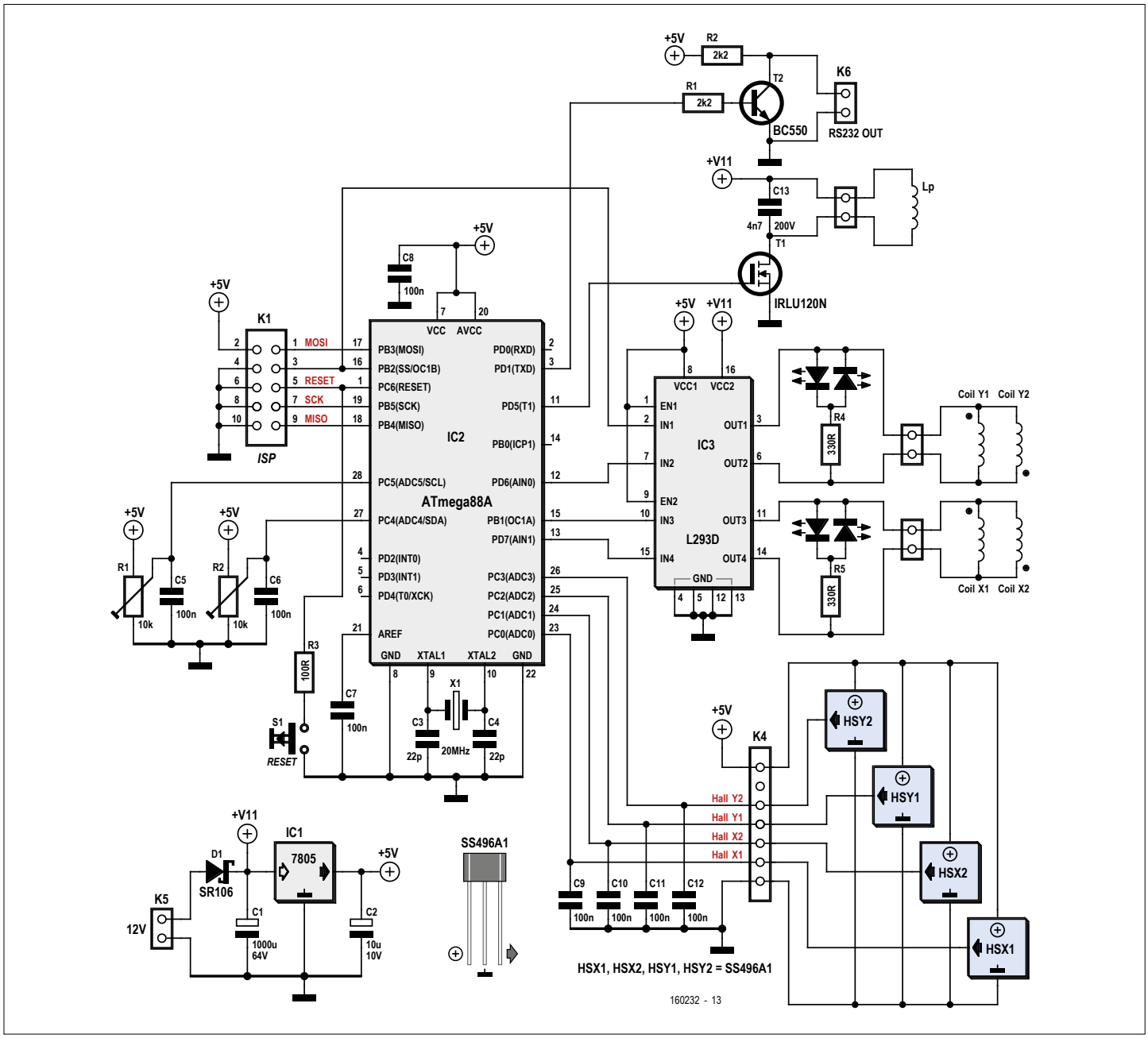


Bild 10. Schaltplan der Regelelektronik.

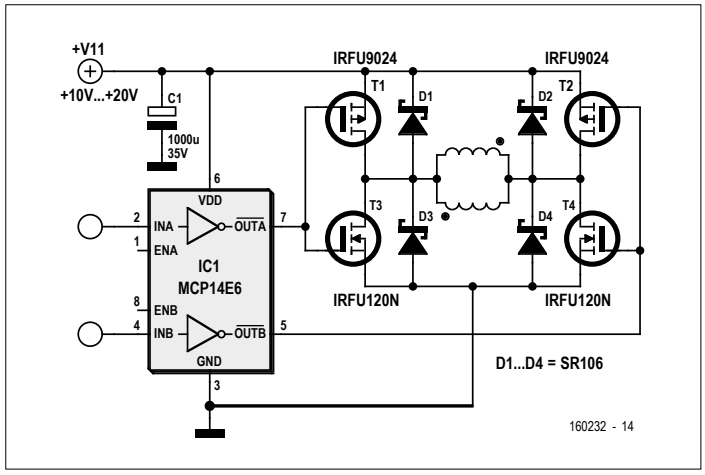


Bild 11. Leistungsbrücke mit MOSFETs.

Bild 12. Elektromagnetplatte der fliegenden Untertasse.

Für experimentierfreudige Leser ist die Software für den ATmega als Gratis-Download erhältlich [4].
Durch händisches Bewegen des Schwebemagneten kann man mit dieser Protokoll-Ausgabe kontrollieren, ob die Sensoren richtig arbeiten und ob der Regler, bei Abweichung des Magneten aus der Mitte, die richtigen Stellsignale erzeugt.
In der ersten Testphase war noch nicht bekannt, wieviel Strom die Elektromagnete benötigen würden. Anstelle des L293D als Brückentreiber wurden daher zwei diskrete Vollbrücken nach dem in **Bild 11** dargestellten Schaltplan verwendet.
Mit dieser Schaltung kann man durchaus einige Ampere durch die Spulen treiben. Für Versuche mit höheren Strömen ersetzt man also besser den L293 durch diese Schaltung (zwei Mal).

Test der Elektromagnetplatte

Der erste Test erfolgt ohne Dauermagnetplatte, so dass nur die Elektromagnete ein Feld erzeugen (**Bild 12**). Zuerst testet man nur den x-Regelkreis, dazu verbindet man nur die x-Spulen sowie alle vier Sensoren mit der Reglerplatine. Das Poti Kp wird halb aufgedreht, das Poti Kd bleibt auf Linksanschlag (Ground). Führt man nun einen Scheibenmagneten mit zum Beispiel 20 mm Durchmesser vorsichtig tastend in der Mitte über die Magnete in x-Richtung, so kann man die Wirkung der Regelung spüren.
Sind die Spulen richtig herum angeschlossen, merkt man außerhalb der Mittelposition eine schwach abstoßende Kraft, die den Magneten in die Mittenposition zurückdrücken will (wenn der Magnet in der richtigen Höhe ist). Bei falscher Polarität wird der Magnet durch die Regelung aus der Mittellage herausgezogen, was wir natürlich gerade nicht wollen. Auf die gleiche Weise sorgt man nun für die richtige Polarität beim Anschluss der y-Spulen. Sind dann beide Spulen angeschlossen, merkt man, wie die Regelung den Magneten in der Mitte hält.
Manche Entwickler bauen die Hallsensoren auch flach an die Spulen wie in **Bild 13** gezeigt. Damit erhielt der Verfasser aber nicht so gute Resultate, da das starke Magnetfeld in z-Richtung die Sensoren bereits fast in die Sättigung brachte.

Zusammenbau und Inbetriebnahme

Beim Zusammenbau ist der Abstand zwischen Magnetplatte und Elektromagnetplatte entscheidend (Maß d in Bild 3). Der Abstand muss so justiert werden, dass der Schwebemagnet genau da schweben will (z-Achse), wo die Elektromagnete die beste x-y-Achsen-Steuerung ermöglichen. Hier muss man notfalls mehrmals verschiedene Abstände ausprobieren. Einen Fehler erkennt man daran, dass die Regelung überhaupt nicht greift.

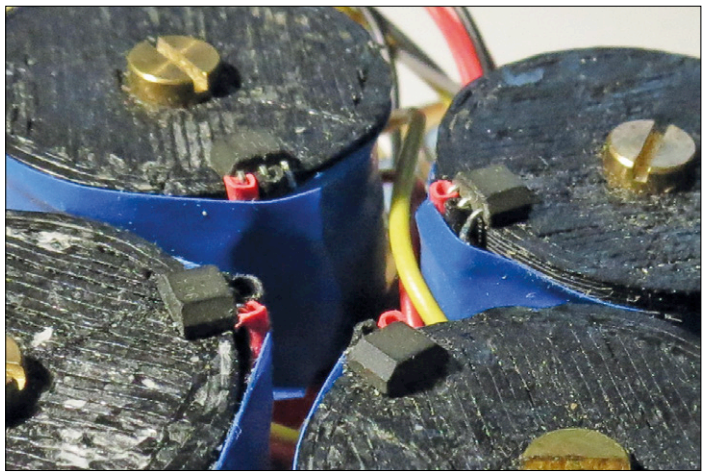


Bild 13. Flach montierte Hallsensoren.

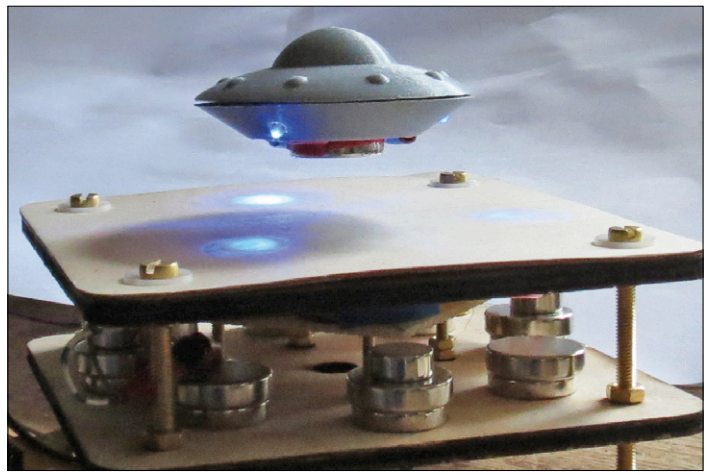


Bild 14. Die fliegende Untertasse.

Nun kommt die eigentliche Inbetriebnahme mit dem Einstellen der Reglerparameter Kp und Kd. Wir starten mit Kp=0 (linker Anschlag) und Kd=0 (linker Anschlag). Den Schwebemagneten hält man nun in der Mitte oberhalb der Elektromagnete dahin, wo er gerade schweben will. Dabei fixiert man nur leicht die x- und y-Richtung. Dann dreht man das Kp-Poti langsam auf. Dabei wird spürbar, wie die Regelung einsetzt und versucht den Magneten in der Mitte zu halten.
Man dreht Kp nun soweit auf, bis diese stabilisierende Wirkung deutlich spürbar ist. Lässt man nun aber den Schwebemagneten los, wird dieser anfangen zu schwingen und dann in Richtung des Ringmagneten klatschen. Um das zu verhindern, braucht man den Differentialanteil des Reglers.
Wir drehen also Kd gerade soweit auf, dass das Schwingen beseitigt wird. Gelingt dies nicht, muss man gegebenenfalls Kp nachjustieren oder auch den Abstand zwischen Magnet- und Elektromagnetplatte. Die allerersten Versuche sind naturgemäß die schwierigsten. Mit der Zeit entwickelt man allerdings eine gewisse Erfahrung.
Der Proportional-/Differential-Regelalgorithmus ist in **Listing 1** zu sehen. Man kann durchaus mit Integerzahlen rechnen; dann ist eine Abtastrate von 1 kHz möglich.
Mit dem 3D-Drucker hat der Verfasser eine „fliegende Untertasse" realisiert, die auch über drei leuchtende LEDs als Scheinwerfer verfügt. In **Bild 14** sieht man die schwebende Untertasse, die in ca. 15 mm Abstand über den Elektromagneten schwebt.

Mehr Abstand

Um einen etwas größeren Schwebeabstand zu erreichen, wurde ein Exemplar mit größerem Durchmesser des Dauermagnetkreises und acht Elektromagneten sowie acht Dauermagneten gebaut (**Bild 15**). Damit wird dann immerhin ein Schwebeabstand von rund 30 mm möglich.

Induktiver Energietransfer

Um induktiv Energie zu übertragen benutzt man oft Konzepte, bei welchen die Primärspule und auch die Sekundärspule mit entsprechenden Kondensatoren in Resonanz gebracht werden. Spannungs- und Stromformen sind dann sinusförmig. Umso überraschter war der Autor, dass die Spannung an der Primärspule der inzwischen eingetroffenen Flyte-Lampe (gemessen mit einer Hilfsspule) so aussah, wie in **Bild 16** dargestellt. Das waren eher aneinandergereihte Sinushalbwellen. Der Autor musste schon tief in seinem Gedächtnis kramen, bis er eine erklärende Vermutung hatte, und die hieß Class-E-Konverter [5].
Ein Class-E-Konverter wird in der Leistungselektronik oft bei sehr hohen Frequenzen eingesetzt. Er ist schematisch wie in **Bild 17** dargestellt aufgebaut. Die Spule L1 ist sehr groß und bewirkt einen näherungsweise konstanten Stromfluss. Lr und Cr bilden ein resonantes Netzwerk, welches die Last R an die Schaltung anschließt. Wenn man eine Gleichstromlast versorgen will, fügt man noch einen Brückengleichrichter hinzu.
Sehr unüblich ist der Kondensator Cp parallel zum Schalter S. Normalerweise stört ein solcher Kondensator in der Leistungselektronik kolossal, weil er beim Schließen des Schalters S kurzgeschlossen wird und seine gespeicherte Energie nutzlos verpufft.
Beim Class-E-Konverter macht man aus der Not eine Tugend. Wir starten die Schaltung wenn Schalter S leitend und Konden-

Listing 1.
Der Proportional/Differential-Regelalgorithmus.

```
// Fehler in x-Richtung berechnen:
errorX=refValueX-HallX;
// Differenz neuer Fehler - alter Fehler:
dErrorX=errorX-prevErrX;
// Steuerwert berechnen:
ctrlX=(errorX*Kp+dErrorX*Kd)/4096 ;
// zur Spule ausgeben:
setCoilX(ctrlX);
// neuer alter Wert = neuer Wert:
prevErrX=errorX;
```

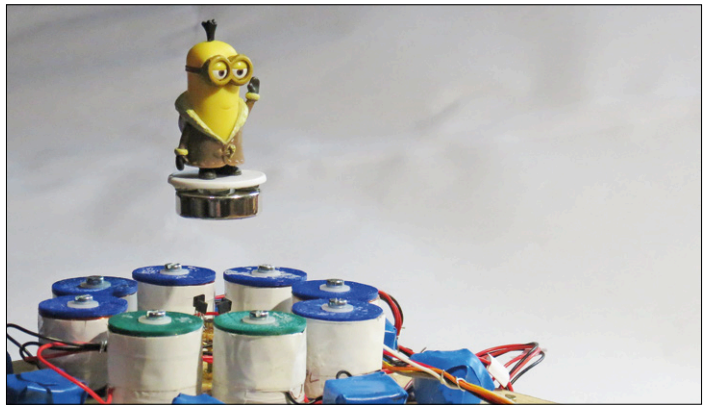


Bild 15. So erreicht man 30 mm Abstand.

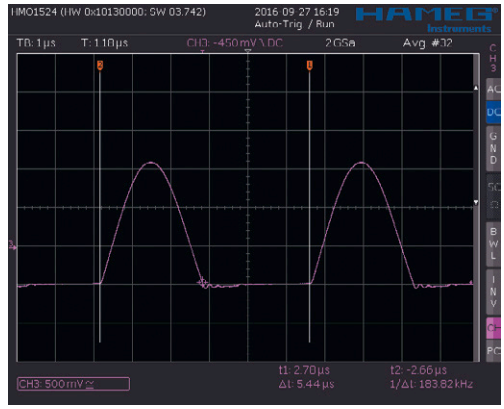


Bild 16. Induzierte Spannung bei der Flyte-Lampe.

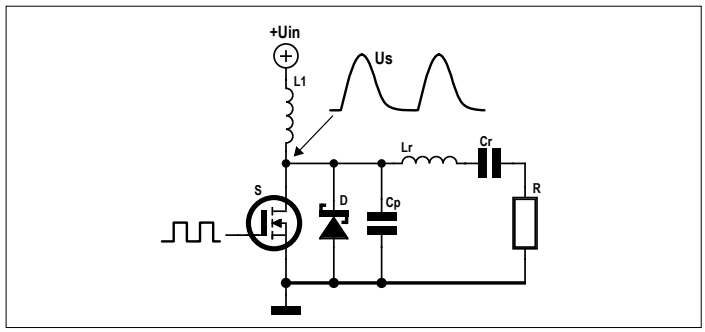


Bild 17. Class-E-Konverter schematisch dargestellt.

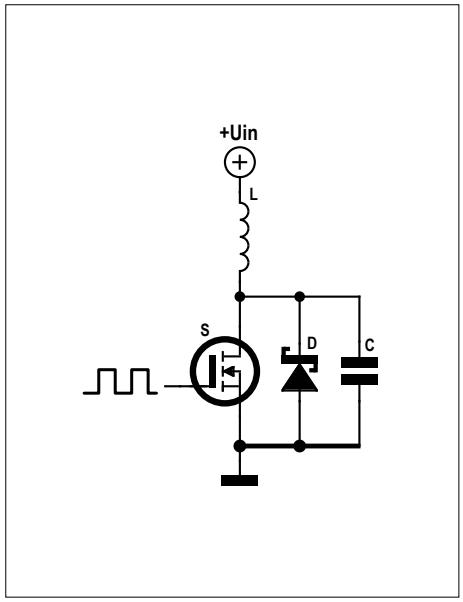


Bild 18. Vereinfachte Class-E-Stufe.

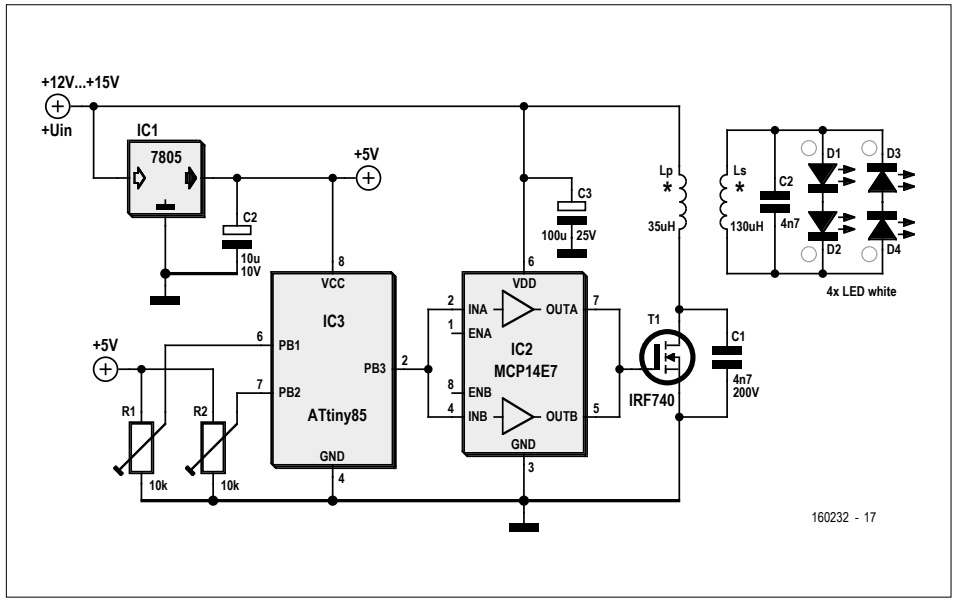


Bild 19. Testschaltung des Class-E-Konverters.

sator Cp ungeladen ist. Bei üblicher Auslegung der Schaltung fließt dann ein positiver Strom (von L1 kommend) durch den Schalter. Nun öffnen wir den Schalter S. Da der Kondensator anfangs ungeladen ist, steigt die Spannung am Schalter nur „langsam“ an, es liegt also in diesem Moment ein „Zero Voltage Switching“ vor.

Das resonante Netzwerk sorgt nun dafür, dass am Schalter S eine Art Sinushalbwelle als Spannungsform erscheint. Wenn die Spannung dann negativ werden will, wird Diode D leitend und die Spannung am Kondensator Cp ist wieder Null. Das hält für eine gewisse Zeit an, innerhalb derer man den Schalter S wieder schließt. Dabei hat man ein „Zero Voltage Switching“ auch beim Einschalten und das Spiel geht von vorne los. Die näheren Details kann man der Literatur [5] entnehmen.
Man kann den Class-E-Konverter nun abspecken und so aufbauen wie in **Bild 18** gezeigt. Spule L bildet mit Kondensator C das resonante Gebilde, das durch den Transistor S zu Schwingungen der in Bild 16 gezeigten Form angeregt wird.

Die Spannung am Drain des Schalttransistors erreicht dabei ein Vielfaches der Versorgungsspannung. Diese Schaltung findet man übrigens auch in induktiven Kochplatten.

Zu Testzwecken wurde eine Schaltung gemäß **Bild 19** aufgebaut. Der ATtiny dient dabei als flexibler PWM-Generator. Das PWM-Signal steuert über einen Gate-Treiber den Schalttransistor T1. Primär- und Sekundärwicklung haben die angegebenen Werte (Lp = 15 Wdg 0,4 mm CuL, 80 mm Durchmesser, Ls = 50 Wdg 0,2 mm CuL, 30 mm Durchmesser), der Koppelfaktor wurde durch Messung zu ungefähr K = 0,08 ermittelt.

In **Bild 20** sieht man den Testaufbau gemäß Bild 19, der problemlos in der Lage ist, vier weiße LEDs in einem Abstand von rund 20 mm zum Leuchten zu bringen. Die gleiche Dimensionierung wurde dann für die schwebende Untertasse übernommen. Da der Class-E-Konverter beim Ein- wie beim Ausschalten ein Zero-Voltage-Switching erreicht, lässt sich der Schalter relativ langsam ein- und ausschalten. Daher kann man sogar auf den

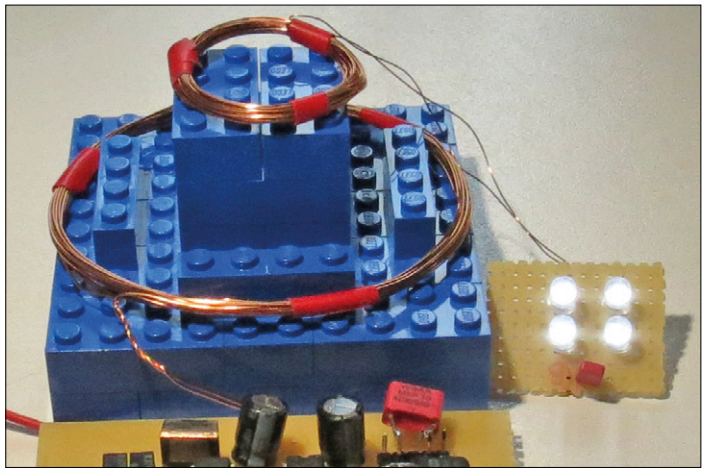


Bild 20. Test des induktiven Energietransfers.

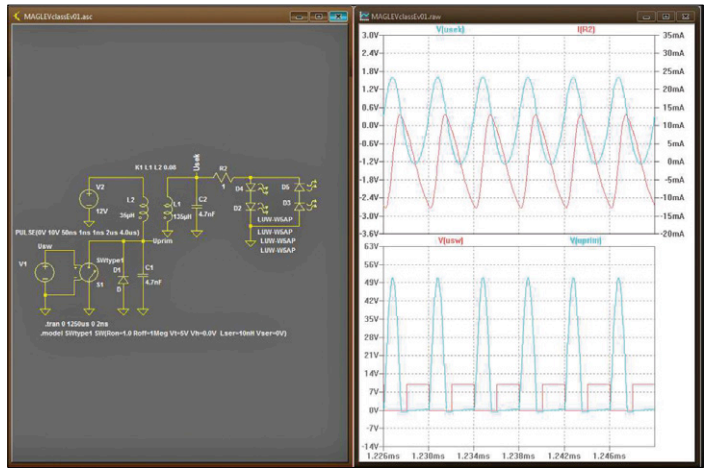


Bild 21. LTspice-Simulation.

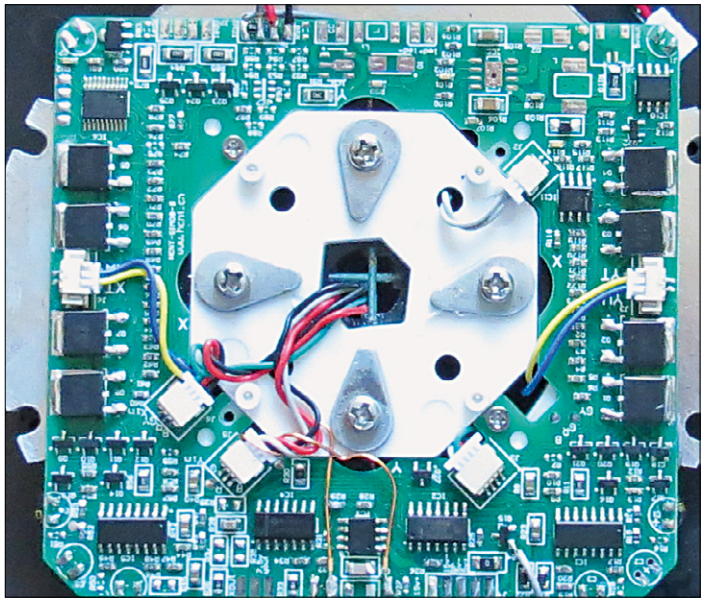


Bild 22. SMD-Seite der Flyte-Platine.

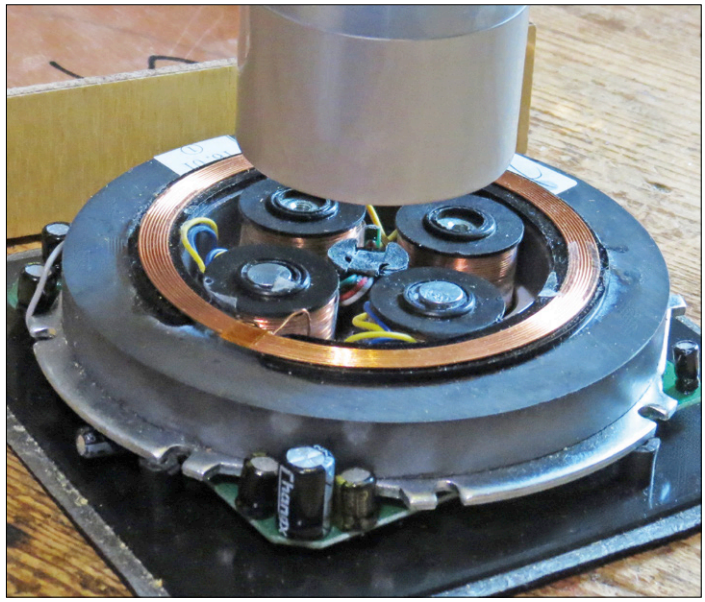


Bild 23. Flyte-Elektronik von oben, mit den Elektromagneten.

Gate-Treiber verzichten und einen Logic-Level-MOSFET direkt vom Mikrocontroller aus ansteuern. In Bild 10 unserer Mikrocontrollerschaltung bilden Lp und C13 den resonanten Kreis, der von T1 geschaltet wird. Als Diode wird die Body-Diode des MOSFETs benutzt. Die Frequenz liegt bei 250 kHz.
Bevor der Konverter gebaut wurde, wurde sein Verhalten mit LTspice simuliert. Die Schaltung und die sich ergebenden Kurvenformen sind in **Bild 21** dargestellt.

Analyse der Flyte-Lampe

Die Lampe ist einfach durch das Aufschrauben von vier Schrauben auseinanderzunehmen. Die SMD-bestückte Unterseite der Platine ist in **Bild 22** dargestellt. Die zwei mal vier „dicken" Transistoren bilden die zwei Vollbrücken zur Ansteuerung der beiden Magnetspulenpaare. Für den drahtlosen Energietransfer ist offensichtlich ein IC namens XKT-510 zuständig [6]. Die Anordnung der Sendespule lässt auf den besprochenen Class-E-Konverter schließen. Auf der Platine befinden sich ein paar nicht identifizierbare ICs und ein Mikrocontroller von ST.

Die Oberseite der Platine sieht man in **Bild 23**.

Das Board trägt den Ringmagneten sowie die vier Elektromagnete. Diese haben einen Eisenkern, der anscheinend justierbar ist (vermutlich um die Lage des schwebenden Magneten zu optimieren). In der Mitte sind wahrscheinlich zwei Hallelemente zur Positionserfassung angebracht. Deutlich sieht man auch die Luftspule mit großem Durchmesser und relativ wenig Windungen. Diese dient dem induktiven Energietransfer. Die ganze Schaltung wird mit 15 V betrieben. Nach unserer Anleitung zum Selbstbau der schwebenden Untertasse ergibt sich bei der Analyse der Flyte-Lampe eigentlich keine Neuigkeit mehr. Ein Patent zu einer derartigen Anordnung findet sich in [2].
Ein CE-Zeichen hat der Autor übrigens vergeblich gesucht. Es ist auch fraglich, ob mit dem so aufgebauten Class-E-Konverter die EMV-Normen erfüllbar sind. EMV-Filterkomponenten fanden sich auf der Platine jedenfalls nicht. ⏮

(160232)

Weblinks

[1] **Flyte-Lampe**: https://flyte.se

[2] **Eberhard Waffenschmidt, Peter Visser, Patent: Floating lamp, WO 2010150207 A1**: www.google.com/patents/WO2010150207A1?cl=de

[3] **Earnshaw, S., On the nature of the molecular forces which regulate the constitution of the luminferous ether., Trans. Camb. Phil. Soc., 7, pp 97-112 (1842)**: www.mit.edu/~kardar/research/seminars/Casimir2010/pdf/EarnshawPaper.pdf

[4] **Software für den ATmega**: www.elektormagazine.de/160232

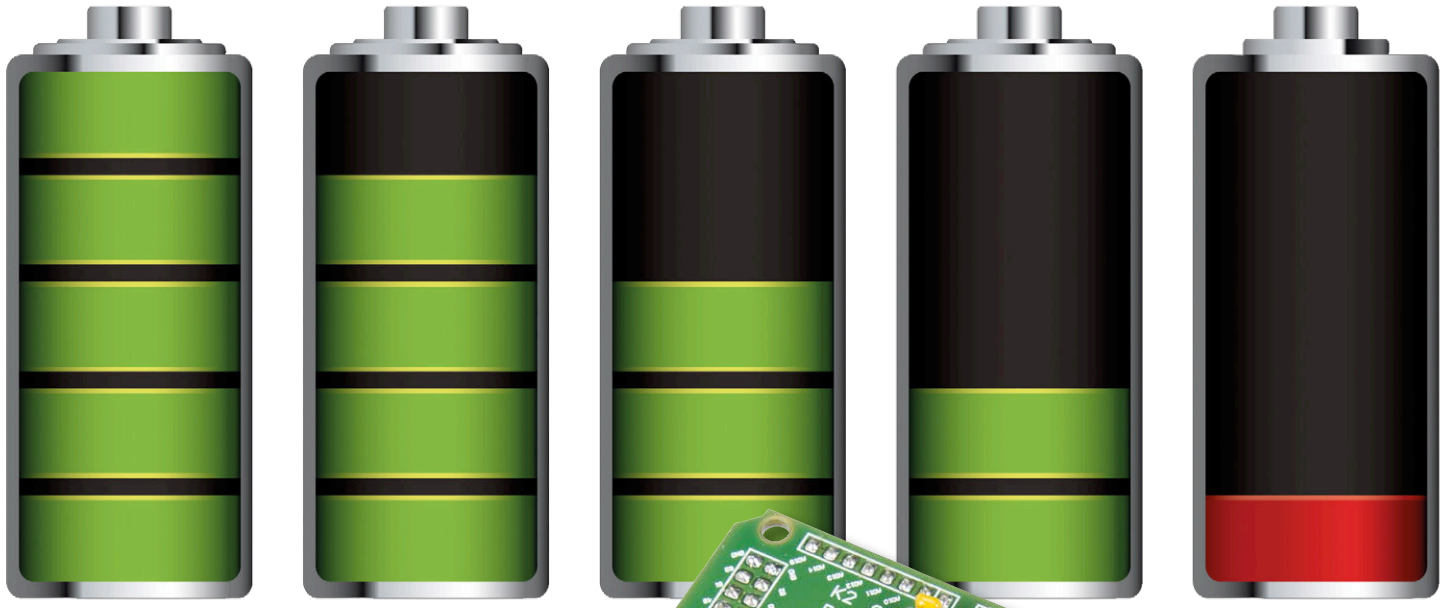
[5] **A high-efficiency Class E inverter – computer model, laboratory measurements and SPICE simulation, Z. Kaczmarczyk**: http://bulletin.pan.pl/(55-4)411.pdf

[6] **XKT-510 IC**: http://elecfreaks.com/store/download/XKT-510.pdf

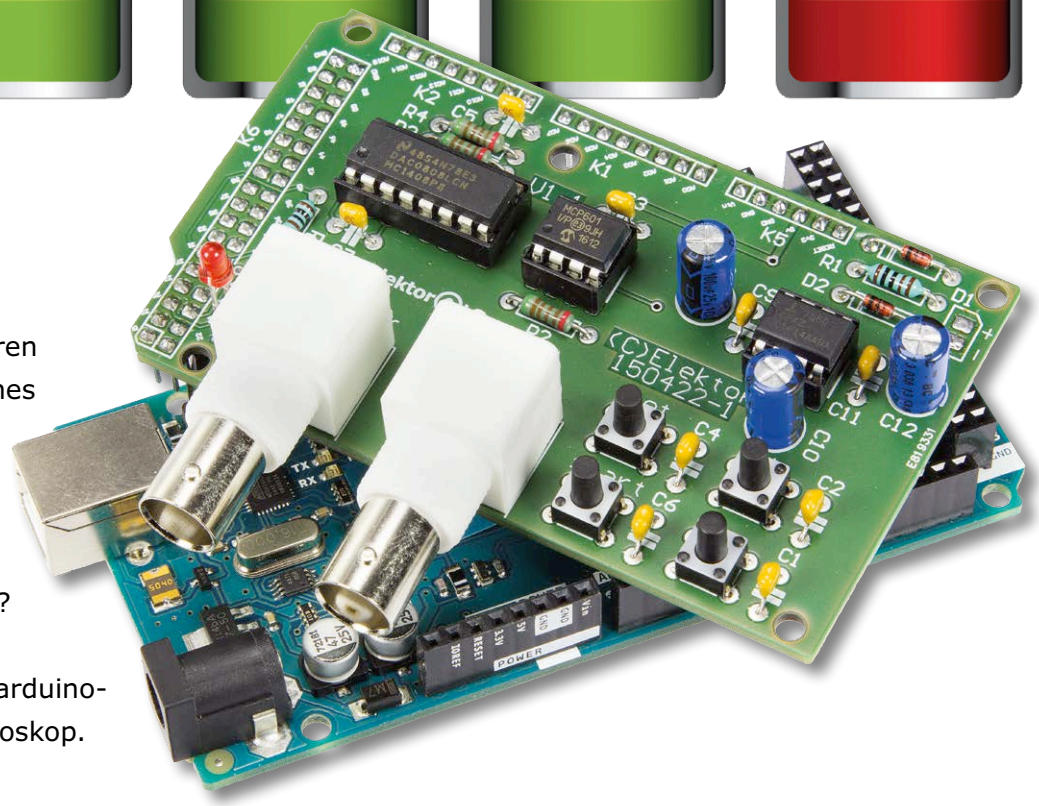
[7] **Ufo-Video**: https://youtu.be/RE9qDVLKxS4

Voltage-Tracker

Langzeitmessungen für Oszilloskope per Arduino-Shield



Von **Kurt Schuster** (D) und
Luc Lemmens (Elektor-Labor)



Wenn man langsame Spannungsverläufe von Sensoren oder etwa die Entladekurve eines Akkus aufzeichnen will, soll man dann daneben sitzen und alle x Sekunden oder Minuten den Wert vom Display eines Multimeters abschreiben? Weniger nervtötend und viel eleganter geht das mit einem arduino-basierten Vorsatz für ein Oszilloskop.

Zwei Fragen motivierten Kurt Schuster zum Start dieses Projekts:
„Taugt dieser Akku noch etwas?“ und *„Lädt dieses Ladegerät den Akku auf richtige Art und Weise?“*
Um sie zu beantworten wäre eine grafische Darstellung des Verlaufs von Lade- und Entladespannung sehr hilfreich. Leider aber stößt hier auch ein modernes Digitaloszilloskop trotz großem Speicher an Grenzen. Die langsamste Horizontalablenkung liegt typischerweise zwischen 1...50 s/div (Sekunden pro Teilstrich). Bei einem Bildschirm mit 12 Teilstrichen horizontal passen so zwischen 12 s und 10 m auf eine Bildschirmseite. Das ist oft viel zu wenig, wenn es um langsame Veränderungen geht, die sich teilweise nur in Stunden abzeichnen. Und auch ein großer Speicher hilft da nicht viel, oder wollen Sie sich durch zig Bildschirme Scrollen? Der richtige optische Eindruck ergibt sich (meistens) nur, wenn der komplette Verlauf in einen Bildschirm passt.
Lösbar ist das Problem natürlich auch mit einem Datenlogger bzw. einem Multimeter mit Logging-Fähigkeiten (dank eingebau-

tem Speicher und/oder Datenübertragung auf einen PC samt passender Software). Aber erstens sind solche Funktionen nicht Standard und die damit ausgestatteten Messgeräte nicht gerade preiswert.
Ergo fiel die Entscheidung, die Zeitbasis des vorhandenen Oszilloskops mit preiswerter Mikrocontroller-Technik gewissermaßen zu spreizen und so mit dem Voltage-Tracker einen geeigneten Vorsatz für Oszilloskope zu entwickeln. Vorab sei gleich verraten: Trotz „digitaler Bedienung“ mit Tasten und Anzeige der Einstellungen auf dem Oszilloskop-Bildschirm eignet sich der Voltage-Tracker nicht nur für modernste Digital-Oszilloskope, sondern funktioniert auch prima mit älteren analogen Oszilloskopen.

Vorüberlegungen

Zunächst stellten sich zwei Fragen: „Wie soll so ein Vorsatz funktionieren?“ und „Welche Bauteile setzt man sinnvollerweise zum Aufbau eines Voltage-Trackers ein?“ Die erste Frage ist recht einfach zu beantworten: Ein Analog/Digital-Umsetzer soll periodisch eine anliegende Spannung abtasten und ein Mikrocontroller soll die sich ergebenden Werte zunächst speichern und später als zeitgerafften Spannungsverlauf wieder analog per Digital/Analog-Wandler periodisch an das Oszilloskop ausgeben. Und wenn schon ein Mikrocontroller vorhanden ist, kann dieser auch für die Bedienung in Form der Abfrage von Tastern und die grafische Anzeige der Einstellungen über entsprechende Spannungspegel des analogen Ausgangssignals sorgen.
Dies führt zur Antwort auf die zweite Frage: Da ein Arduino preiswert ist und die etablierten Entwicklungswerkzeuge in Form der Arduino-IDE kostenlos zu Verfügung stehen, braucht man das Rad nicht schon wieder neu zu erfinden, sondern verwendet einfach ein Arduino-Board als fertige Grundlage. Die erforderliche Zusatz-Hardware wird dann als aufsteckbares Shield ausgeführt, was die Anforderungen an die zugehörige Platine in Grenzen hält.
Ein Mikrocontroller hat für diese Zwecke in der Regel genügend RAM, denn man muss ja nur so viele Datenpunkte speichern können, wie horizontale Punkte bei einem Digital-Oszilloskop vorhanden sind – der Rest ist überflüssiger Luxus. Kurt Schuster verwendete einen Arduino Mega 2560 mit immerhin 8 KB RAM. Erforderlich sind aber für Daten und das von ihm geschriebene Programm lediglich 1,8 KB RAM. Folglich tut es fast jedes Arduino-Board und sogar eines des Typs Uno, solange es qua Rechenleistung und Timing mit 16 MHz betrieben werden kann.
Wenn man die Sache ganz spartanisch ausführen würde, dann bräuchte man außer dem Mikrocontroller bzw. einem Arduino-Board prinzipiell nichts weiter. Ein ATmega-Mikrocontroller verfügt ja immer schon über mindestens einen „analogen Eingang“ hinter dem sich ein recht schneller ADC verbirgt. Doch während der analoge Eingang ganz brauchbar ist, kann man einen analogen Ausgang de facto nur mit einem digitalen Ausgang realisieren, der im PCM-Modus bzw. pulsbreitenmoduliert betrieben wird. Idealer und präziser ist da schon ein „richtiger“ DAC als eigener Chip, der ein angelegtes Byte direkt in eine analoge Spannung umwandeln kann. Mit einem nachfolgendem Rail-to-Rail-Opamp als Puffer bzw. niederohmige „Endstufe“ kann man ein Ausgangssignal erzeugen, das sauber auf Masse bezogen ist.

Bauteile und Schaltung

Man könnte nun einfach einen hypermodernen und quasi „alleskönnenden“ DAC samt Tastern etc. auf eine geeignete Platine packen und fertig wäre das Shield. Doch erstens sind solche Chips nicht ganz preiswert und zweitens heute immer SMD-behaust. Da die Shield-Platine genug Platz hat, bietet es sich zwecks einfacheren Aufbaus und geringeren Anforderungen an die Lötkünste des Nachbauers an, nur bedrahtete Bauteile einzusetzen. Glücklicherweise gibt es einen bewährten, preiswerten, alten aber noch immer sehr gut erhältlichen DAC des Typs DAC0808, der noch wie ein richtiges IC aussieht und nicht durch den Lufthauch beim Ausatmen von der Platine fliegt, wenn man ihn einlöten möchte. Insgesamt ist der Aufbau der Schaltung also sehr einfach, zumal es von Elektor sogar eine fertige Platine (**Bild 1**, siehe Stückliste) dazu gibt. Man muss lediglich die Firmware auf

Stückliste

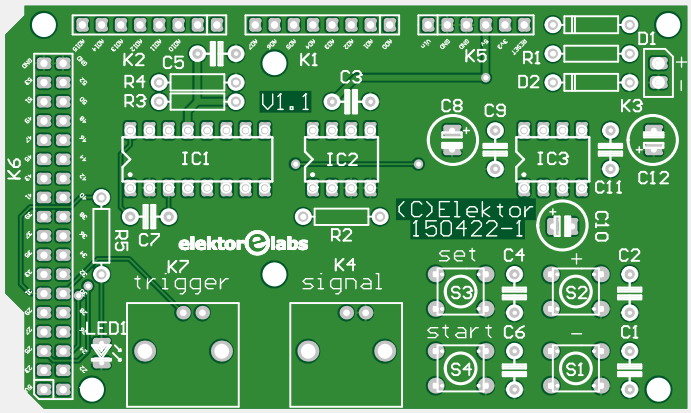


Bild 1. Vom Elektor-Labor entwickelte Platine, die auch im Shop erhältlich ist.

Halbleiter:
D1, D2 = 1N4148
LED1 = LED, rot, 3 mm
IC1 = DAC0808LCN, DIP16
IC2 = MCP601-I/P, DIP8
IC3 = ICL7660CPAZ, DIP8

Außerdem:
K1, K2 = 1x8-pol. Stiftleiste
K3 = 1x2-pol. Stiftleiste
K4, K7 = BNC-Buchse, Platinenmontage, abgewinkelt
K5 = 1x6-pol. Stiftleiste
K6 = 2x18-pol. Stiftleiste
S1..S4 = 1-pol. Taster, 6x6 mm, Platinenmontage
IC-Fassungen für IC1..IC3
Zwei Coax-Kabel mit BNC-Steckern
Platine 150422-1 im Elektor-Shop
Arduino Mega 140566-93 im Elektor-Shop

Achtung: Alle Stiftleisten auf Unterseite bestücken!

Widerstände:
R1, R5 = 1 k, 5%, ¼ W
R2..R4 = 5k11, 1%, ¼ W, Metallfilm

Kondensatoren:
C1..C7, C9, C11 100 n, 50 V, X7R, RM 2/10’’
C8, C10, C12 = 100 µ, 25 V, radial, RM 1/10’’

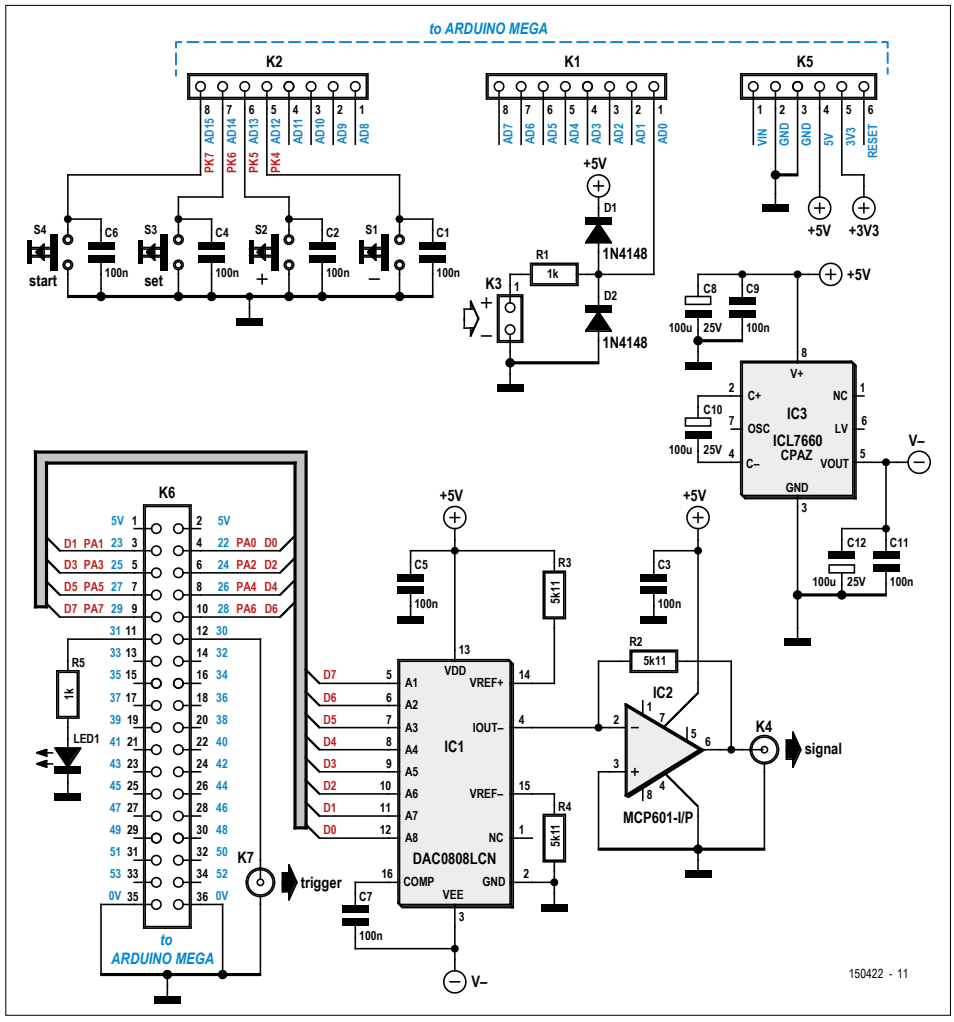


Bild 2. Die Schaltung des Arduino-Shields fällt sehr übersichtlich aus.

das Arduino-Board laden. Die zugehörige Sketch-Datei steht zusammen mit den Platinen-Layouts als kostenloser Download auf der Elektor-Webseite zu diesem Artikel [1] zur Verfügung.

In der Schaltung von **Bild 2** ist zu sehen, wie dieser DAC alias IC1 von Port A (**K6**) des ATmega angesteuert wird. IC2 ist der Ausgangspuffer. Gewählt wurde hier ein Rail2rail-Opamp von Microchip. Die Elektronik eines Arduino wird ja gewöhnlich mit 5 V versorgt, die auch dem Shield zur Verfügung stehen. Die nötige, jedoch nur schwach belastete negative Hilfsspannung für den DAC wird vom Spannungskonverter IC3 erzeugt. Wie so oft ist die Schaltung eines Arduino-Shields sehr übersichtlich. Es sind lediglich noch vier Taster für die Bedienung sowie eine LED erforderlich. Die zu messende Spannung wird an **K3** des Shields angelegt. Damit nun der analoge Eingang und damit der Mikrocontroller des Arduino-Boards nicht durch verpolte oder zu große Eingangsspannungen beschädigt wird, sind mit R1 ein Schutzwiderstand und zwei Dioden eingefügt, die das Zuviel oder Verkehrte an Spannung ableiten und auf unschädliche Werte begrenzen.

Weil der ADC des Mikrocontrollers verwendet wird, ist der messbare Spannungsbereich auf 0...5 V festgelegt. Das passt sehr gut z.B. für eine Lion-Akkuzelle, denn hier kommen je nach Typ maximal 4,6 V vor. Will man ganze Akkupacks oder z.B. einen 12-V-Akku messen, schaltet man einfach einen Spannungsteiler vor, der auch durchaus im 100-kΩ-Bereich liegen kann, denn der analoge Eingang ist mit rund 100 MΩ sehr hochohmig.

Die Schaltung wird wie schon erwähnt direkt vom Arduino-Board versorgt. Dieses wiederum bezieht entweder 5 V via USB (z.B. direkt von einem Digital-Oszilloskop) oder 7,5...12 V über ein typisches Arduino-kompatibles Netzteil.

Das Shield wird über zwei Signal-Leitungen an das Oszilloskop angeschlossen,

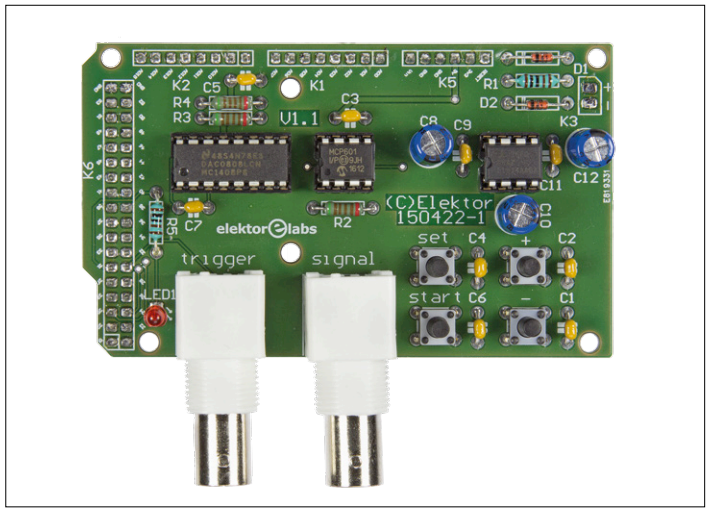


Bild 3. Die fertig bestückte Platine ist bereit zum Aufstecken auf ein Arduino-Board.

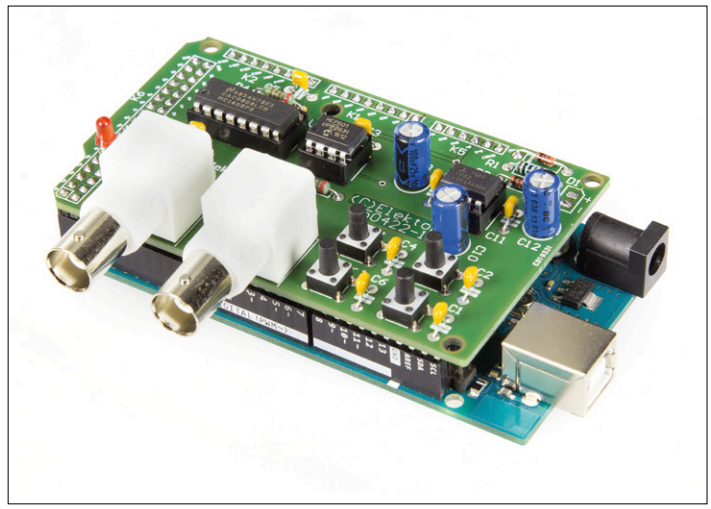


Bild 4. Das Sandwich aus Shield + Arduino ergibt einen kompakten Voltage-Tracker.

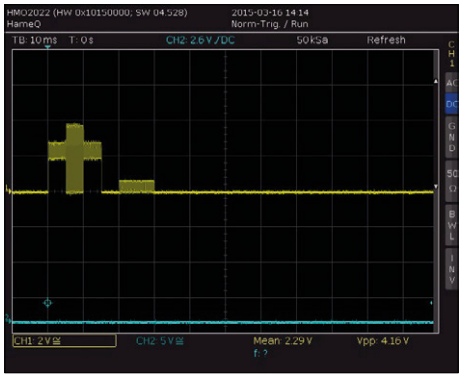


Bild 5. Anzeige bei der Einstellung von **time**. Zu sehen ist ein „t“ plus eine Stufe der Rampe, was 1 m/div bedeutet.

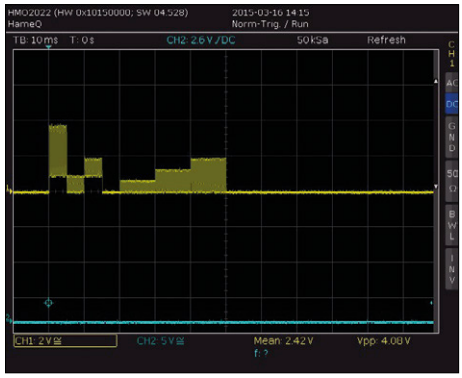


Bild 6. Die Einstellung von **level**. Zu sehen ist ein „l“ plus drei Stufen der Rampe, was einen Offset von 3 V zur Folge hat.

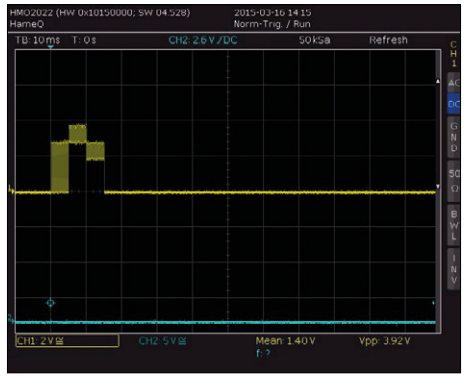


Bild 7. Aufzeichnungsart **roll**. Zu sehen ist ein „r“ ohne Rampenstufe, wodurch nur ein Plot aufgezeichnet wird.

wovon eine Leitung das analoge Signal (**K4**) und die andere das Trigger-Signal (**K7**) führt. Für beide Ausgänge sind BNC-Buchsen vorgesehen. Man benötigt also zwei handelsübliche Coax-Leitungen mit BNC-Steckern an beiden Enden, um das Shield mit einem Oszilloskop zu verbinden. **Bild 3** zeigt das fertig bestückte Shield. In **Bild 4** ist zu sehen, wie ein Arduino-Board mit aufgestecktem Shield eine kompakte Einheit bildet.

Betrieb und Einstellung

Da der Mikrocontroller die Messungen im eigenen Speicher ablegt, muss während der möglicherweise stundenlangen Messungen das Oszilloskop nicht eingeschaltet sein. Dies gilt aber nicht während der Einstellung der diversen Optionen, denn sonst sieht man nichts, wenn man die Taster betätigt. Ist die Hardware fertig, Arduino mit dem nötigen Sketch versehen und das Shield aufgesteckt, kann man den Voltage-Tracker per BNC-Kabel mit dem Oszilloskop verbinden (Daten- und Triggerleitung) und die Stromversorgung einschalten.

Auf dem Shield befinden sich vier Taster. Je einer davon ist (wird) mit „**-**“ und „**+**“ beschriftet. Mit diesen fährt man durch die Stufen einer einzustellenden Option, die man mit S3 (**set**) wählt. Ein kurzer Druck auf S4 (**start**) startet oder stoppt einen Messverlauf und ein längerer Druck (>2 s) löscht den bisher aufgezeichneten Verlauf. Soweit so einfach. Doch wie sieht man, was gerade eingestellt wird? Hierzu steuert der Arduino den DAC so, dass sich auf dem Bildschirm des Oszilloskops der Anfangsbuchstabe der jeweiligen Einstellung plus eine Rampe ergeben, deren Stufen der gerade gewählten Option entsprechen.

Das Oszilloskop stellt man auf einen Triggerpegel von 1...4 V ein oder aktiviert ansonsten die Trigger-Automatik. Die Zeitbasis wird auf 10 ms/div und die Vertikalverstärkung auf 2 V/div eingestellt sowie geprüft, ob das Oszilloskop den Teilerfaktor der Prüfspitzen auf 1:1 gesetzt hat. Ist alles richtig eingestellt, sollte ein kleines „**t**“ für die Zeiteinstellung sichtbar sein und eine Rampenstufe dahinter angezeigt werden.

Die Breite der Stufe sollte genau 1 div betragen. Stimmt die Breite nicht, sollte man nochmals die Zeitbasis-Einstellungen des Oszilloskops überprüfen. Hat das Oszilloskop keine horizontale Zwölfterteilung, passt man den Tracker so an: Die SET-Taste so oft drücken bis ein kleines „**n**“ (Anzahl der „divs“) angezeigt wird. Mit den Tasten „**+**“ und „**-**“ wird der Tracker im Bereich 10...16 div an den Oszilloskop-Schirm angepasst (siehe auch weiter unten im Text). Nach der Einstellung die SET-Taste betätigen, bis wieder „**t**“ erscheint. Jetzt kann man die anderen Parameter anhand der nachfolgenden Menüs an den Messzweck anpassen. Die auf dem Shield befindliche LED und die LED des Arduinos bleiben während der Einstellungen dunkel. Sie blinken während der Aufzeichnung und leuchten dauernd, wenn die Aufzeichnung beendet ist. Nachfolgend noch die fünf möglichen Einstellungs-Optionen samt den jeweiligen Stufen:

Tabelle 1: Bei der Zeitbasis gibt es insgesamt sieben Stufen (siehe **Bild 5**). Die Dauer bezieht sich auf einen Oszilloskopschirm mit Zwölfterteilung. Da die Rampe im Bild nur eine Stufe hat, ist eine Horizontalauflösung von 1 m/div eingestellt.

Tabelle 2: Hier wird der Offset in fünf Stufen eingestellt (siehe **Bild 6**), ab der eine Spannung registriert wird. Man muss etwas aufpassen, denn beim im Bild dargestellten Level von 3 V hat man nur noch einen maximalen Hub von 2 V, um im maximalen Messbereich von 5 V des Mikrocontrollers zu bleiben (siehe auch Vertikaleinstellung).

Tabelle 3: Mit der in **Bild 7** dargestellten Stufe 0 wird die Datenaufzeichnung bei einem vollen Bildschirm abgebro-

Tabelle 1. Zeitbasis: time

Stufe	Zeit/div	Dauer
1	1 m	12 m
2	2 m	24 m
3	5 m	1 h
4	10 m	2 h
5	20 m	4 h
6	30 m	6 h
7	1 h	12 h

Tabelle 2. Level-Einstellung: level

Stufe	Offset
0	0 V
1	1 V
2	2 V
3	3 V
4	4 V

Tabelle 3. Aufzeichnung: roll

Stufe	Rolling
0	aus
1	ein

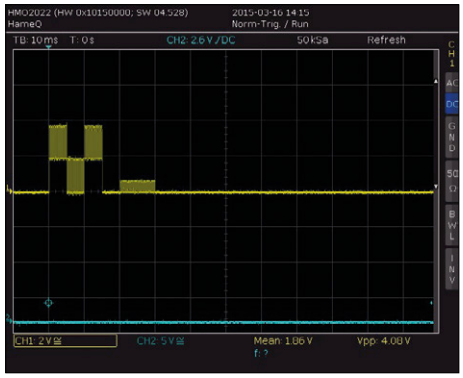


Bild 8. Bei der Einstellung der **y-Skala** muss man auch den eingestellten Level beachten. Zu sehen ist ein „y" mit einer Rampenstufe, was einen Messbereich von 2,5 V bedeutet.

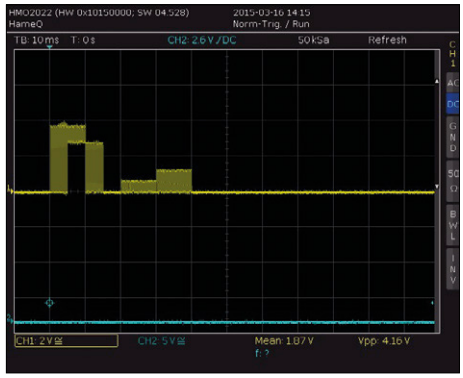


Bild 9. Anzeige bei der Einstellung von **number** zur Anpassung an die horizontale Teilung des Oszilloskops. Zu sehen ist ein „n" mit zwei Rampenstufen, was einem Bildschirm mit Zwölferteilung entspricht.

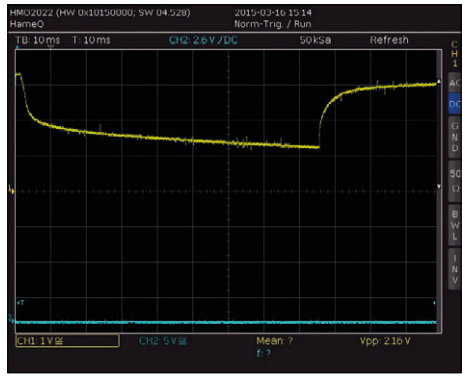


Bild 10. Angezeigter Verlauf am **Beispiel** der Entladekurve einer Li-Ion-Akkuzelle. Dargestellte Dauer = 12 Minuten.

chen. Bei Stufe 1 hingegen gibt es eine rollende Darstellung, bei der bei jedem neuen Datenpunkt rechts auf der linken Seite ein Datenpunkt gelöscht wird.

Tabelle 4: Die Vertikaleinstellung ist etwas komplizierter. In **Bild 8** ist Stufe 1 eingestellt, was einer zweifachen Verstärkung entspricht. Da der maximale darstellbare Spannungshub jetzt 2,5 V beträgt, kann beim Level nur ein maximaler Wert von 2 V eingestellt werden, wenn man Clipping = Überschreiten des Messbereichs vermeiden will. Mit Level = 2 V ergibt sich ein Messbereich von 2...4,5 V. Mit Level = 0 V hat man einen nichtverschobenen, auf Masse bezogenen Messbereich von 0...2,5 V.

Tabelle 5: Nicht jeder Bildschirm eines Oszilloskops weist horizontal die gleiche Teilung auf. Um die Zeitbasis des Voltage-Trackers an das angeschlossene Oszilloskop anzupassen, gibt man hier die Anzahl der Teilstriche ein. Dies hat auch Auswirkungen auf die registrierbare Zeitdauer. In **Bild 9** ist mit Stufe 2 eine übliche Zwölferteilung eingestellt. In der höchsten Stufe ergeben sich 16 Teilstriche, was einer maximalen Registrierdauer von 16 Stunden entspricht.

Beispiel

In **Bild 10** wird die Entladekurve einer (gebrauchten) Li-Ion-Akkuzelle dargestellt. Zuerst wird entladen und bei Unterschreitung von 3,3 V wird die Entladung gestoppt. Man sieht schön, wie sich die Spannung in der Recovery-Zeit anschließend wieder erholt. Verwendet wurde ein Oszilloskop mit Zwölfer-Teilung. Die Einstellungen beim Voltage-Tracker waren 1 m/div, was einer Gesamtdauer von zwölf Minuten entspricht. Level war auf 3 V und die Vertikaleinstellung auf ¼ eingestellt. Zusammen mit einer Vertikalverstärkung am Oszilloskop von 1 V/div ergibt sich bei diesem Plot eine vertikale Auflösung von 0,25 V/div.

Schlussbemerkungen

Der Bau eines Arduino-Shields mit lediglich bedrahteten Bauteilen ist sicherlich eine leichte Übung. Zusammen mit einem preiswerten Arduino ergibt sich ein ebenso preiswerter Voltage-Tracker zur praxistauglichen Aufzeichnung und Darstellung langsamer Spannungsverläufe mit Hilfe eines Oszilloskops. Die Bedienung ist einfach, da der Bildschirm des Oszilloskops als Display für die Einstellungen genutzt wird.

Der Voltage-Tracker ist nicht vergesslich: Alle Einstellungen werden im EEPROM des Arduino-Mikrocontrollers gespeichert und stehen beim nächsten Einschalten sofort zur Verfügung. ⏮

(150422)

Tabelle 4. Vertikaleinstellung: y-Skala

Stufe	Y-Osz.	Bereich	Level max.
0	1/4	1,25 V	4 V
1	1/2	2,50 V	2 V
2	3/4	3,75 V	1 V
3	1/1	5,00 V	

Tabelle 5. Bildschirm-Teilung: number

Stufe	Teilung
0	10
1	11
2	12
3	13
4	14
5	15
6	16

Weblinks

[1] www.elektormagazine.de/150442

[2] www.elektormagazine.com/labs/voltage-tracker-for-oscilloscope

Über den Autor:

Kurt Schuster arbeitet als Hardware- und Software-Entwickler mit besonderem Schwerpunkt auf AVR-Assembler. Feedback und Fragen bitte an: qrt@qland.de.

16.-18. Februar Heidelberg
MakerCon 2017

MakerCon – From Maker to Market

Die Konferenz für Maker

Heidelberg, 16.–18.2.2017

Jetzt anmelden

- **Wege zur Professionalisierung des eigenen Maker-Projekts**
- **Ein FabLab gründen: Modelle und Fördermöglichkeiten**
- **Den Maker-Spirit ins Unternehmen holen: Projektmanager und Innovationsbeauftragten lernen von der Maker-Szene**

Zielgruppen

- Ambitionierte Maker und Communities
- Organisatoren von MakerSpaces oder FabLabs
- Innovations- und Technikbeauftragte von Unternehmen
- Anbieter von Hardware und Dienstleistungen sowie Startups

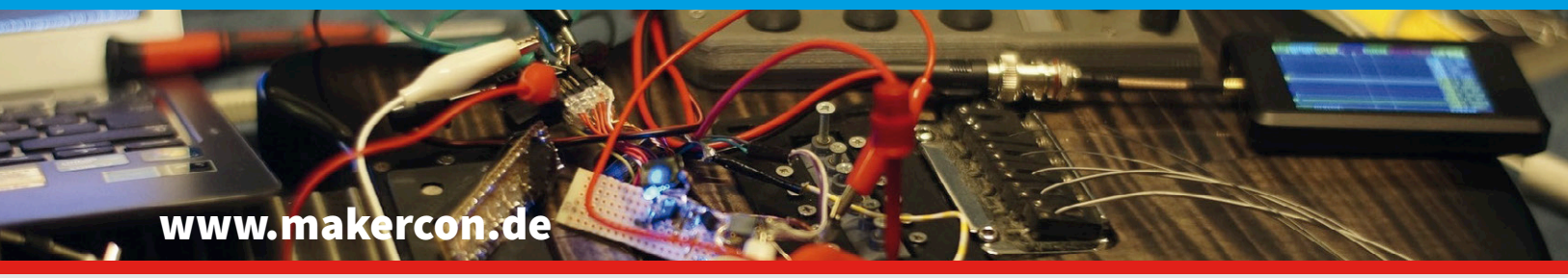
Keynote

Mario Lukas:
From Maker to Market – wie ich vom Spielzeug-Dealer zum Open-Source-Maker wurde

Dale Dougherty:
Makers and Open Innovation Ecosystems

Auszug aus dem Programm

- Materialien und deren Verarbeitung
- Prototyping im professionellen Umfeld
- FPGAs für Maker
- Konstruieren mit Stahl – wie legt man los?
- Wie Prototyping die Innovationskultur im Großkonzern fördert
- BMW @ MakerSpace München: Ein DAX-Konzern geht neue Wege
- Wie gründe ich einen Makerspace? Ein Erfahrungsbericht
- Makerspace Evolution



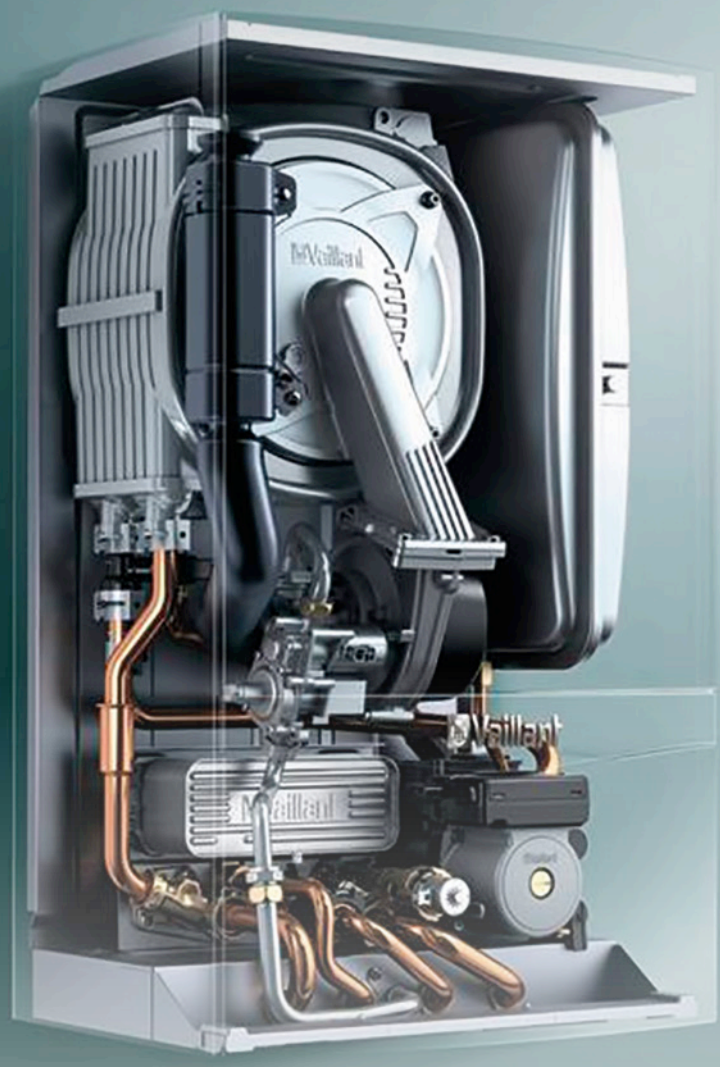
Goldsponsor: CONRAD

Veranstalter: Make: heise Developer dpunkt.verlag

Heizkesselmonitor mit ESP8266

Haustechnik für die Energiewende

Von **Walter Trojan**



Die Heizung ist und bleibt ein Kostenfaktor für die Haushaltskasse. Deshalb kann es nicht schaden, dem Heizkessel etwas genauer auf die Rohre zu schauen und vorhandene Optimierungspotentiale zu nutzen. Der Heizkessel-Monitor erfasst die wesentlichen Betriebsparameter, zeigt sie auf einem kleinen Bildschirm an und loggt sie für spätere Auswertungen auf einem USB-Stick. Da die Daten in das WLAN eingespeist werden, sind sie im ganzen Haus oder sogar im Internet abrufbar.

In der Übersicht (**Bild 1**) ist die Client-Server-Architektur des Heizkessel-Monitors gut zu erkennen. Der Mess-Server besteht aus einem autonom arbeitenden ESP8266 auf einem ESP-201-Modul. Die Arbeit mit dem Chip wurde von Elektor schon mehrfach ausführlich beschrieben [1][2]. Der Mess-Server verfügt außerdem über fünf Temperatur-Sensoren DS18B20 und ein Mikrofon (!) zur galvanisch entkoppelten Erfassung der Brenneraktivitäten. Der ESP ist als Station konfiguriert und überträgt im Intervall von 2 s mittels TCP/IP die Messwerte ins WLAN.

Um den Entwicklungsaufwand zu minimieren, besteht der Anzeige-Client aus einem preiswerten STM32F429 Discovery-Board [3]. Es ist bereits mit einem kleinen Touchscreen ausgestattet und verfügt über einen USB-Anschluss für das Datenlogging auf einem Speicherstick. Für die WLAN-Ankopplung ist ebenfalls ein ESP8266 zuständig, der auf einem ESP-01-Modul montiert ist. Auch dieser ESP arbeitet als Station und empfängt über TCP/IP die Nachrichten des Mess-Servers. Diese werden über eine serielle Verbindung unmittelbar zum STM32 geleitet, der die Messwerte verdichtet, anzeigt und auf dem USB-Stick speichert.

Auch an Zeitstempel ist gedacht: Da das Discovery-Board keine Pufferbatterie für die Echtzeituhr besitzt und ich mir die Umstellung von Sommer/Winterzeit sowie die Berücksichtigung von Schaltjahren ersparen wollte, habe ich zu diesem Zweck einfach das Internet angezapft und hole mir die aktuelle Zeit im NTP-Protokoll von einem Zeitservice.

Auf dem USB-Stick wird für jeden Tag eine separate Datei erstellt, welche die Daten im Minutentakt enthält. Das Format ist so gewählt, dass im Nachgang eine einfache Analyse zum Beispiel mit Excel durchgeführt werden kann. Der Bildschirm des Discovery-Boards zeigt die jeweils aktuellen Messwerte. Nach diesem Überblick nun der Einstieg in die Details.

Der Mess-Server...

Der Mess-Server basiert auf dem WLAN-Chip ESP8266, der neben seinen dedizierten WLAN-Schaltkreisen über einen leistungsfähigen 32-Bit-Prozessor verfügt, der mit 80 MHz beziehungsweise 160 MHz getaktet wird. Von der Prozessorleistung werden lediglich 20 % für die WLAN-Funktionen benötigt, der Rest kann für zusätzliche Aktivitäten genutzt werden.

Die chipinterne MCU Tensilica L106 verfügt neben 32+80 KB RAM über nützliche Peripherie, die über 17 Leitungen von außen zugänglich ist. Diese können als GPIOs, UART-, SPI-, I2C- oder PWM-Schnittstellen geschaltet werden. Die Firmware wird in einem externen Flash-Speicher untergebracht, der über eine leistungsfähige SDIO-Schnittstelle (vier Daten-, zwei Steuerleitungen) angesteuert wird. Der Strombedarf von circa 70 mA (maximal 170 mA) kann mit den Power Management Units im Sleep-Modus auf etwa 10 µA reduziert werden. Es gibt zahlreiche Module mit dem ESP8266, angefangen mit dem ESP-01 mit zwei GPIOs. Beim Mess-Server wird ein ESP-201 mit umfangreicherer Peripherie eingesetzt.

Folgende Aufgaben werden von diesem Modul erledigt:

- Zyklische Messung der Temperaturen von Vor- und Rücklauf der Heizung, der äußeren Umgebung sowie Warmwasser-Erhitzung und –Entnahme.
- Überwachung der Brenneraktivitäten mit einem Mikrofon.
- Bereitstellung der Messwerte im WLAN.

Die übersichtliche Schaltung mit dem zentralen ESP8266 ist in **Bild 2** dargestellt. Die Temperatur-Sensoren sind über die GPIOs 2, 4, 5, 12 und 14 anschlossen, wobei die DQ-Leitungen mit Pullup-Widerständen versehen sind. Die Spannungsversorgung der Sensoren erfolgt mit V_{dd} = 3,3 V.
Das verwendete kleine Mikrofon enthält bereits einen Vorverstärker – versorgt mit 7,5 V – und liefert beim Lärm des Brenners eine Ausgangsspannung von ungefähr 5 V_{ss}, die von D1 und D2 gleichgerichtet wird, den Transistor T1 durchschaltet und so an GPIO 13 die Brenneraktivitäten signalisiert.

Zur Programmierung des ESP8266 sind die Anschlüsse TX, RX, RST und GPIO 0 auf die Stiftleiste K1 geführt. Im Betrieb sind RST, CH_PD, GPIO 0 auf High und GPIO 15 auf Low geschaltet. Zur Debug- und Betriebsanzeige steuert GPIO 15 die LED 1 an. Die unkritische Spannungsversorgung erfolgt mit einem 7,5-V-Steckernetzteil. D3 ist die Verpolschutz-Diode, der Spannungsregler LM3940 reduziert und stabilisiert die Spannung auf 3,3 V.

Die Lösung im Überblick

Der Heizkessel-Monitor besteht aus zwei Komponenten, dem Mess-Server und dem Anzeige/Logging-Client. Beide sind mittels TCP-Verbindung über das WLAN verbunden. Folgende Parameter werden gemessen und angezeigt:

- Vorlauf-Temperatur des Heizwassers
- Rücklauf-Temperatur des Heizwassers
- Außen-Temperatur
- Temperatur für Warmwassererhitzung
- Warmwasser-Entnahmetemperatur
- Brenneraktivitäten

Bei der Erfassung der Messwerte werden keinerlei Eingriffe in die Elektrik des Heizkessels vorgenommen und das 230-V-Netz nicht tangiert. Das macht die Entwicklung für sämtliche Heizkesseltypen tauglich und schützt den Anwender vor gefährlichen Stromschlägen.
Weitere Aufnahmen meiner Installation sind bei Elektor /Labs [9] zu sehen.

...und seine Sensoren

Die Temperaturen werden von den DS18B20-Sensoren ermittelt, die im Bereich von -10...+85 °C mit einer Genauigkeit von ±0,5 °C messen und die Werte im 1-Wire-Protokoll digital ausgeben. Obwohl es das Protokoll ermöglicht, alle Sensoren parallel an einen Draht zu schalten, habe ich doch jedem Sensor einen eigenen GPIO spendiert. Das vereinfacht die Firmware und die Identifizierung der Bauteile entfällt.

Die Temperatur-Sensoren sind in robusten Stahlgehäusen mit gutem Temperatur-Übergang erhältlich (Aliexpress [4]). Sie lassen sich, wie **Bild 3** zeigt, gut mit den Heizungsrohren verbinden. Die Sensoren werden mit einem wärmeleitenden Kleber an den Rohren von Vorlauf und Rücklauf fixiert, mit ebenfalls wärmeleitender Kupferpaste satt umhüllt, mit einem Abschnitt längs halbierten Kupferrohrs abgedeckt und mit einer Edelstahlrohrschelle fixiert. Der Wärmeübergang

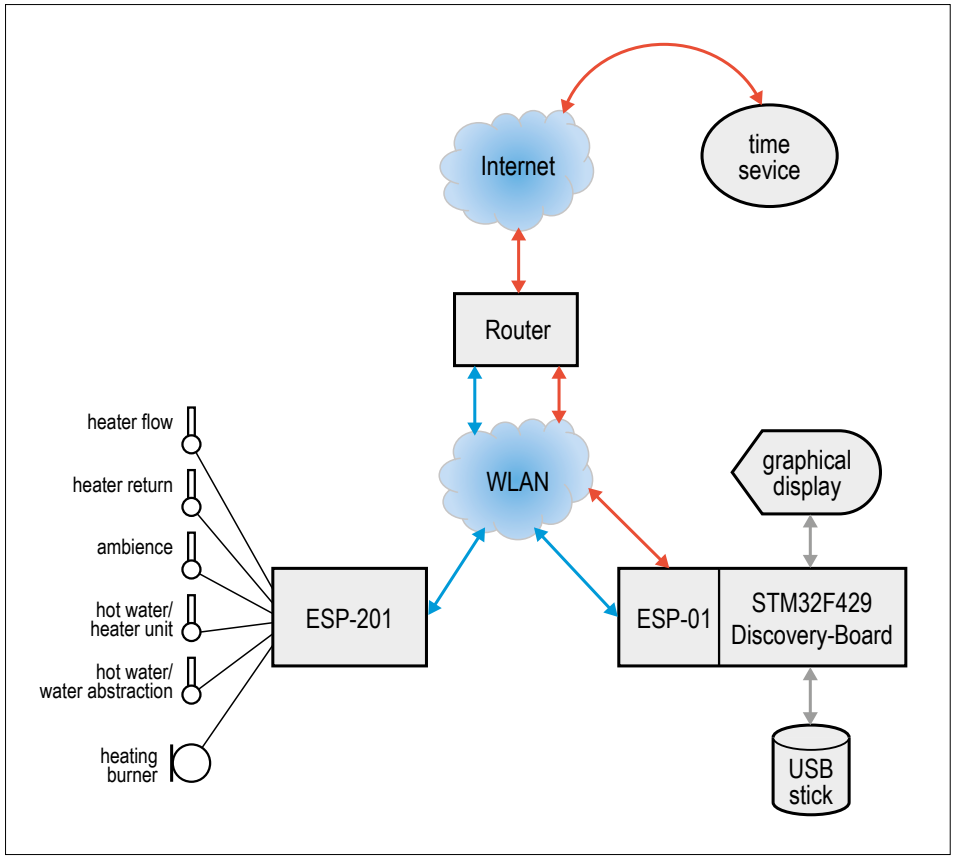


Bild 1. Übersicht des Projekts „Heizkessel-Monitor“.

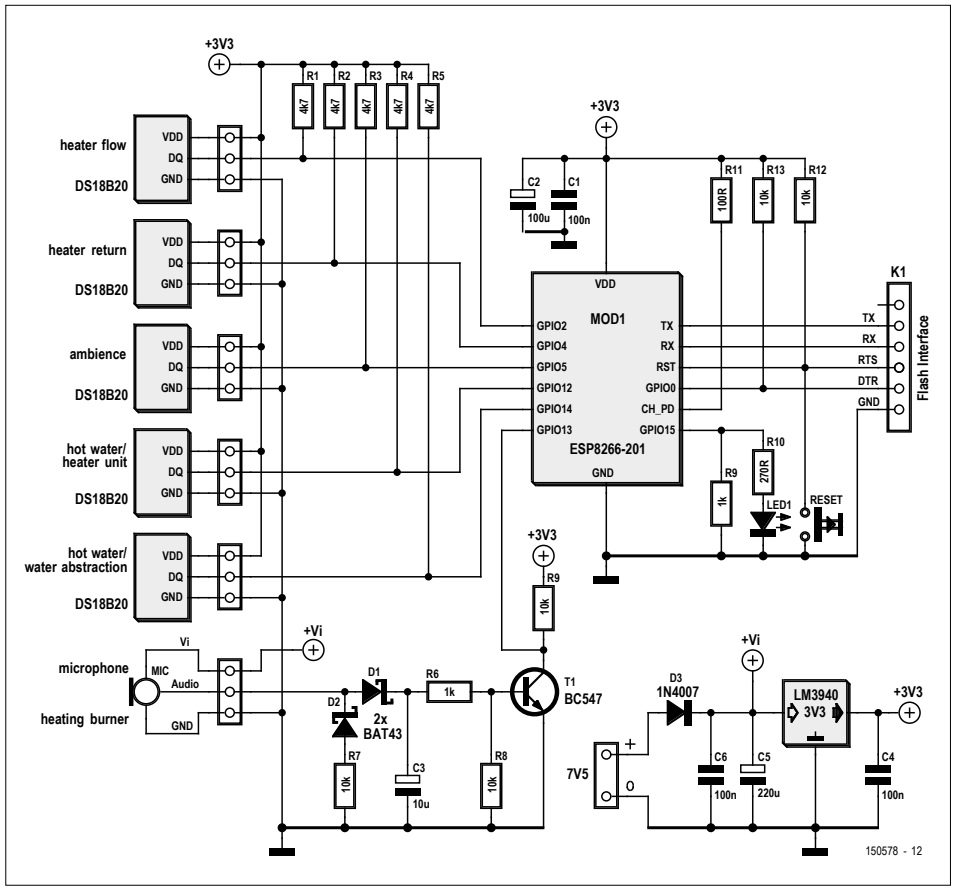


Bild 2. Die Schaltung des Mess-Servers mit dem zentralen ESP8266.

ist nicht perfekt, ich habe mit einem analogen Messgerät eine Abweichung von etwa 2 °C ermittelt. Eine Wärmeisolation der Messstellen (mit Kaiflex oder ähnlichem) würde das Resultat sicher verbessern. Das kleine Mikrofon habe ich unmittelbar an der Luftklappe des Brenners aufgehängt, damit es dessen Aktivitäten zuverlässig erfassen kann.

freeRTOS statt „Bare-Metal“

Wenn man bisher nur „Bare-Metal“-Projekte, also unmittelbar auf der MCU-Hardware aufgesetzte Firmware programmiert hat, hat man zwar die volle Kontrolle, aber nur wenig Komfort. Zwar ist der Umstieg auf ein komfortables Environment mit Real-Time-Operating-System mit einer gewissen Umstellung verbunden, die Einarbeitung macht sich aber bezahlt. Hier wird zur Steuerung des Mess-Servers das Echtzeit-Betriebssystem freeRTOS eingesetzt, das Espressif [5] auch in seinen SDKs nutzt.

Bei der Programmierung sind folgende Dinge zu beachten:

- Verzögerungsschleifen mit CPU-Belastung sind zu vermeiden und durch (Soft-) Timer-Konstrukte zu ersetzen.
- Timer-Variablen sollte man nie lokal, sondern statisch definieren, was eigentlich selbstverständlich ist.
- Eine Anwenderfunktion darf nicht länger als 15 ms dauern, da sonst konkurrierende WLAN-Funktionen behindert werden oder der Watchdog-Timer einen Restart auslöst.

Zur Programmierung der Firmware habe ich das „Unofficial Development Kit for Espressif ESP8266“ von Cherts [6] mit Eclipse und dem MinGW-C-Compiler unter Windows benutzt.

Die Struktur der Firmware ist in **Bild 4** zu sehen. Die Gesamtsteuerung übernimmt wie bei allen C-Programmen eine main-Funktion. Sie läuft im Hintergrund und steuert unter anderem alle WLAN-Funktionen.

Nach dem Start des ESP8266 wird die Funktion user-init einmal aufgerufen. In dieser Funktion werden

- die Parameter für die Peripherie gesetzt, in diesem Fall für UART und GPIOs,
- eine Init-Callback und ein Event-Handler gestartet,
- eine Systick-Funktion für eigene Soft-Timer (hier 10-ms-Tick) aktiviert
- eine eigene Hauptschleife, hier workloop genannt, im Intervall von 10 ms aufgerufen.

Jetzt kann man bereits das Konstruktionsprinzip gut erkennen, es handelt sich um zahlreiche Callback-Routinen. Mit der Init-Callback wird gemeldet,

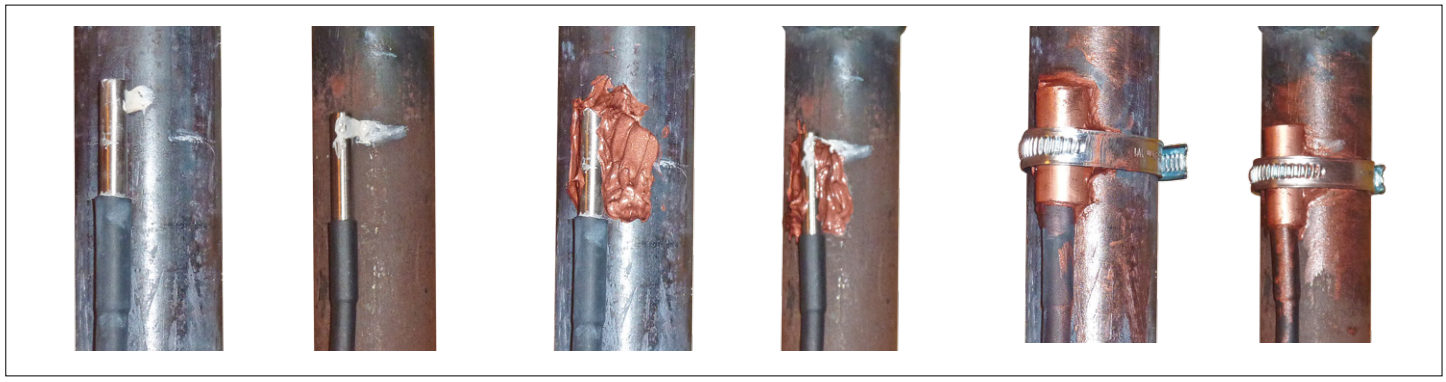


Bild 3. Zur Montage der Temperatursensoren.

wenn der ESP einsatzbereit ist, also alle WLAN-Einheiten korrekt arbeiten. Auch der Event-Handler ist eine Callback-Routine, die unter anderem die hier nützlichen Ereignisse meldet:

- EVENT_STAMODE_CONNECTED : ESP ist als Station mit dem Router verbunden
- EVENT_STAMODE_DISCONNECTED : Verbindung mit dem Router getrennt
- EVENT_STAMODE_GOT_IP : ESP hat vom Router eine IP-Adresse erhalten

In keiner Callback-Routine findet eine nennenswerte Verarbeitung statt, es werden lediglich in globalen Variablen entsprechende Signale gesetzt.
Die Verarbeitung findet in den von der workloop aufgerufenen DoApp-State-Maschinen statt, wobei DoApp2 die WLAN-spezifischen Aktivitäten übernimmt und DoApp3 sich um die LED-Ansteuerung kümmert. Die Auswertung der Sensoren findet in DoApp1 statt.

Workloop ist hier keine klassische Endlosschleife, sondern endet nach dem sequentiellen Aufruf der oben genannten Apps, aber nicht, ohne sich vorher beim Betriebssystem mit einer kurzen Verzögerung selber wieder aufzurufen. So wird der Ablauf nicht blockiert und die Systemfunktionen erhalten die Steuerung zurück. Eine Endlosschleife unter RTOS sieht aus wie in **Listing 1.**
Im Laufe der Verarbeitung werden noch weitere Callback-Routinen initialisiert:

- `tcpclient_connect_cb` : Ein Client hat sich eingeloggt
- `tcpclient_disconnect_cb` : Ein Client hat sich ausgeloggt
- `tcp_recv_cb` : Eine TCP-Message wurde empfangen
- `tcp_sent_cb` : Eine TCP-Message wurde erfolgreich gesendet

Auch in diesen Funktionen wird lediglich das Ereignis in einer Variablen signalisiert und so zum Beispiel an DoApp2 weitergegeben.

Die DoApps arbeiten im kooperativen Multithreading. Nach der globalen Initialisierung in user_int wird die Hauptschleife workloop aktiviert, in der die Threads sequentiell aufgerufen werden. Diese sind als State-Maschinen ausgeprägt, in der Funktionen in Abhängigkeit einer Statusvariablen abgearbeitet werden. Ist die Funktion erfolgreich abgearbeitet, verändert sie die switch-Variable, so dass beim nächsten Aufruf gemäß Aufgabenstellung eine andere an die Reihe kommt.
Die Firmware

- initialisiert in der user_init GPIOs, UART, workloop- und systick-Schleifen sowie die init_done- und event-Callback-Funktionen. Nach erfolgreicher Initialisierung startet die workloop mit zyklischen Aufrufen der DoApps.
- DoApp2 verbindet den ESP mit dem WLAN und startet einen TCP-Server.
- Hat sich ein Client eingeloggt, startet DoApp1 mit den Messungen. Dabei wird zuerst die Temperaturmessung bei den Sensoren gestartet und nach ungefähr 2 s werden die Werte abgefragt. Der Zustand des Brenners wird über eine einfache GPIO-Abfrage ermittelt.
- Ist die Messung komplett, überträgt DoApp2 die Daten zum Client.
- Im Hintergrund überwacht DoApp3 den Betrieb und zeigt den Zustand mit der grünen LED an: „aus“ = keine Verbindung, „blinkt langsam“ = Verbindung zum WLAN OK, „an“ = Client hat sich angeschaltet, „kurz aus“ = Übertragung zum Client.

Aufgrund des „handgemachten“ Multithreadings lassen sich die einzelnen Aufgaben relativ unabhängig programmieren und testen. Weitere Details über die

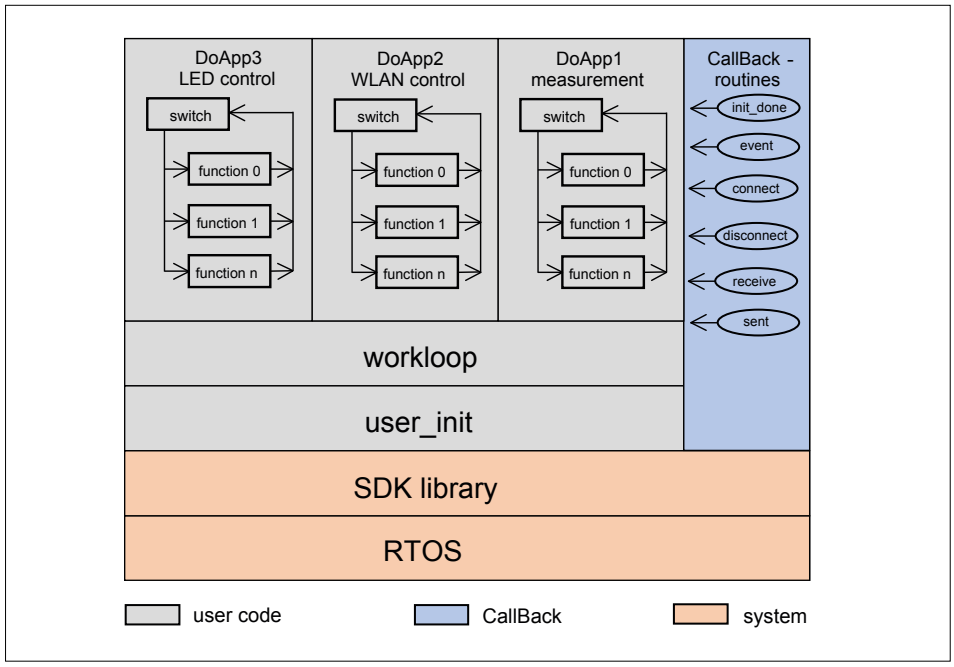


Bild 4. Die Struktur der Firmware des Mess-Servers.

Listing 1. Endlosschleife in RTOS.

```
LOCAL void ICACHE_FLASH_ATTR workloop(void *arg)
{
    os_timer_disarm(&Wloop);        // Schaltet Wloop-Timer aus

    DoApp1();                       // State-Machine Messen
    DoApp2();                       // State-Machine WiFi
    DoApp3();                       // State-Machine LEDs
    os_timer_setfn(&Wloop, (os_timer_func_t *)workloop, NULL);
                                    // Ruft wookloop wieder auf
    os_timer_arm(&Wloop, 10, 0);    // nach einer Pause von 10 ms
}
```

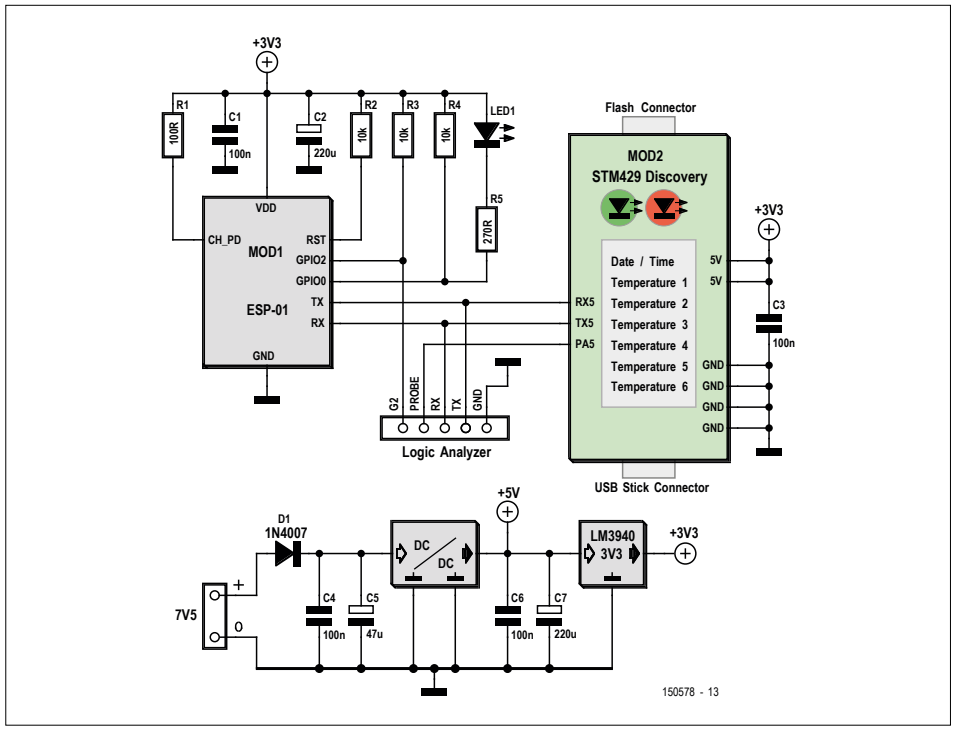


Bild 5. Die Schaltung des Anzeige/Logging-Clients.

Firmware können Sie den Quellcodes auf der Elektor-Projektseite [6] entnehmen. So, die Messdaten stehen nun im WLAN bereit, jetzt ist nur noch ein Abnehmer erforderlich.

Der mobile Anzeige/ Logging-Client

Damit man die Messwerte überall im Hause und auch im Internet abgreifen kann, sind die Anzeige- und Logging-Funktionen in einer separaten Einheit untergebracht. Sie

- empfangen und verdichten WLAN-Daten vom Mess-Server,
- zeigen die aktuellen Werte auf einem grafischen Display an und
- speichern sie auf einem USB-Stick in einem Excel-verträglichen Format.

Zusätzlich werden Uhrzeit und Datum mit einem Internet-Zeit-Service synchronisiert. Bei der Realisierung dieser Lösung habe ich bewährte und preisgünstige Module eingesetzt, das STM32F429 Discovery Kit für Anzeige und Logging und das ESP8266-Modul ESP-01 für die WLAN-Kopplung. Dadurch wird, wie **Bild 5** beweist, die Schaltung sehr übersichtlich und wenig komplex. Beide Systeme sind zur Übertragung von Daten und Befehlen über verdrillte RX/TX-Leitungen mit 115.000 bit/s gekoppelt. Zum Betrieb des ESP sind GPIO-, RST- und CH_PD-Pins über Pullup-Widerstände auf V_{dd}-Potential gelegt. GPIO 0 steuert eine LED und signalisiert den Betriebszustand.

Mit Firmware wird der ESP von einem externen Programmiergerät versorgt. Wichtig ist der gut bemessene Elko C2, der die hohen Stromspitzen des ESP bei der WLAN-Kommunikation puffert. Damit die Funktionen komfortabel überprüft werden können, sind einige wichtige Leitungen auf einen 10-poligen Stecker herausgeführt, an dem ein Logik-Analysator angeschlossen werden kann: Über RX/TX lässt sich die Kommunikation überwachen, die Pins GPIO 2 und PA5 liefern flexibel einsetzbare Probesignale an kritischen Software-Punkten. Sehr hilfreich bei der Entwicklung!
Das STM32F429-Discovery-Kit umfasst alle nötigen Komponenten: Programmieradapter, 2,4-Zoll-Touch-Display, USB-Anschluss und LEDs. Das Board ist mit der MCU STM32F429 mit 180 MHz Takt ausgestattet, die „on chip" über 2 MB Flash sowie 256 KB RAM verfügt. Zusätzlich stehen 64 Mbit SRAM bereit, etwa zur Speicherung von Grafiken. Über den Flash-Verbinder in Form einer USB-Minibuchse wird die Firmware eingespielt. An eine zweite USB-Minibuchse wird der USB-Speicherstick angeschlossen. Die grüne LED zeigt den Betriebszustand des Boards; die rote signalisiert, ob ein Stick angeschlossen ist (an) oder ob momentan auf den Stick geschrieben wird (blinkt). Genau wie beim Mess-Server ist die Stromversorgung unkritisch. Über die Verpolungs-Schutzdiode D1 werden 7,5 V eines Steckernetzteils zugeführt, von einem Gleichspannungswandler auf 5 V für das Disco-Board und von einem LM3940 nochmals auf 3,3 V für den STM32 heruntergeregelt.

Anzeigen und Loggen im Teamwork

Die Firmware für den Anzeige/Logging-Client gliedert sich wie auch die Hardware in zwei Teile, nämlich das Programm für die WLAN-Kommunikation mit dem ESP8266

Listing 2. Zeit aus dem Internet.

```
void InitNTP(void)
{
    sntp_setservername(0, "ptbtime1.ptb.de");
                                    // set server 0 by domain name
    sntp_setservername(1, "ptbtime2.ptb.de");
                                    // set server 1 by domain name
    sntp_setservername(2, "ptbtime3.ptb.de");
                                    // set server 2 by domain name
    sntp_set_timezone(1);           // MEZ
    sntp_init();                    // NTP initialisieren
}

...

current_stamp = sntp_get_current_timestamp();
                                    // aktuelle Zeit holen
```

und die Routinen für Anzeige/Logging auf dem Disco-Board.

Wie nicht anders zu erwarten, fällt die Firmware für den ESP sehr ähnlich wie die des Mess-Servers aus. Es gibt allerdings nur zwei State-Maschinen, DoApp2 ist für das WLAN und DoApp3 für die Steuerung der grünen LED zuständig. Anstatt zyklisch zu senden, lauscht der Client auf eintreffende Meldungen und gibt diese über die RX/TX-Verbindung an das Discoboard weiter. Da dieses keine Pufferbatterie für die RTC hat, besorgt der ESP die exakte Uhrzeit samt Datum aus dem Internet, damit die STM32-Clock synchronisiert werden kann. Eine gute Quelle dafür ist die Physikalisch-Technische Bundesanstalt (PTB) in Braunschweig, die drei Zeit-Services bereitstellt. Ptbtime1 wird als Quelle genutzt, die anderen Server sind Ersatzquellen, falls bei ptbtime1 Probleme auftreten (**Listing 2**).

Die Firmware auf dem Discoboard hat vier State-Maschinen, die allerdings nicht von einem RTOS, sondern bare-metal von der endlosen Hauptschleife aktiviert werden. Die Entwicklung erfolgte in C mittels CooCox, einer freien IDE für ARM-Cortex-CPUs. Diese Entwicklungsumgebung ist allerdings etwas in die Jahre gekommen, für Neuentwicklungen mit dem STM32 sollte heutzutage TrueStudio von Atollic oder System Workbench mit CubeMX zum Einsatz kommen.

Die einzelnen State-Maschinen führen folgende Aktivitäten durch:

- *DoApp1:*
  - Empfängt (vom ESP) die Messwerte und konvertiert sie ins Anzeigeformat.
  - Empfängt Uhrzeit/Datum und setzt die Realtime-Clock.

- *DoApp2:*
  - Bedient den Touchscreen (Start/Stop).
  - Zeigt die aktuellen Messwerte auf dem Display an.

- *DoApp3:*
  - Zeigt im Abstand von zwei Sekunden Uhrzeit und Datum auf dem Display an.
  - Verwaltet den USB-Stick und präpariert darauf eine Datei pro Tag.
  - Bildet die Mittelmesswerte für eine Minute und schreibt sie in die Datei.

- *DoApp9:*
  - Steuert die LEDs auf dem Discoboard.

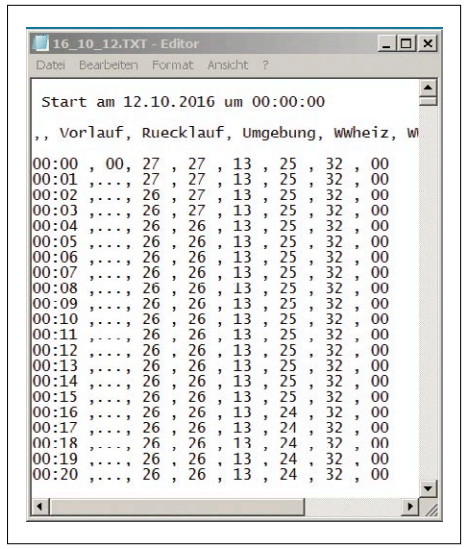


Bild 6. CSV-Dateiformat für die Daten auf dem USB-Stick.

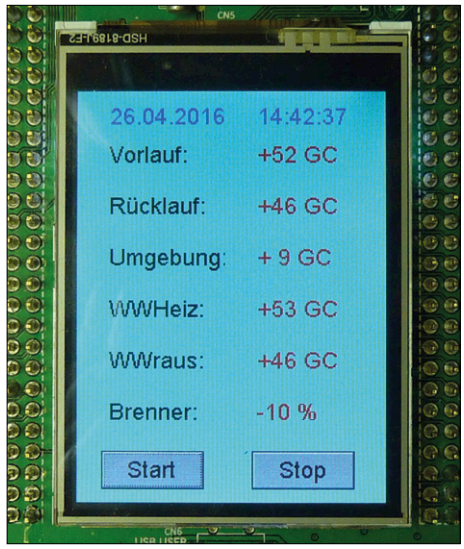


Bild 7. Die empfangenen Daten auf dem Display des STM32-Boards.

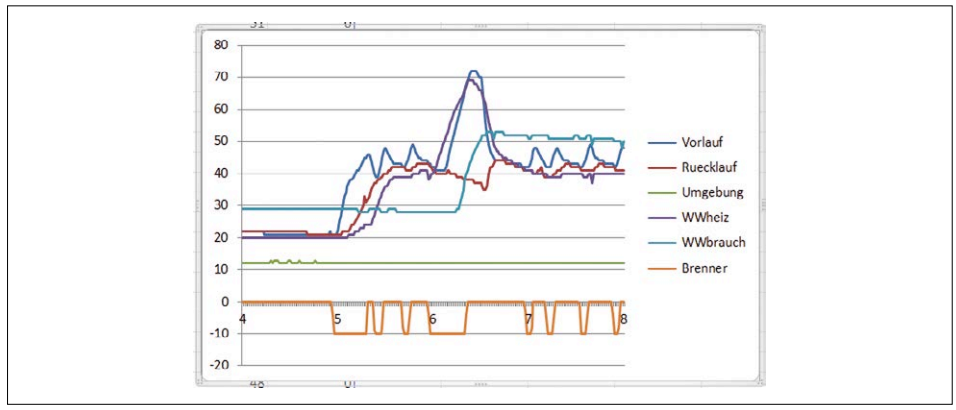


Bild 8. So kann ein Analyse-Diagramm in einer Tabellenkalkulation aussehen.

Bild 6 zeigt das Format der Daten auf dem USB-Stick. Für jeden Tag wird eine separate Textdatei (im CSV-Format) mit der Bezeichnung Jahr_Monat_Tag.TXT erstellt. In ihr werden die verdichteten Messwerte von 00:00 bis 23:59 Uhr gespeichert. CSV-Dateien können leicht in einer Tabellenkalkulation wie Excel oder OpenCalc verarbeitet werden.

Jeder Minuteneintrag enthält neben der Uhrzeit die fünf Temperaturwerte sowie die prozentuale Brenneraktivität. Der Brennerwert ist allerdings skaliert, wobei -10 = 100 % bedeutet. Dieses macht die spätere grafische Auswertung übersichtlicher, da die Brennerwerte nicht im Temperaturbereich, sondern darunter dargestellt werden.

Der Bildschirm des Discoboards in **Bild 7** zeigt neben Datum und Uhrzeit die Messwerte. Die übersichtliche Grafik wurde mit dem emWinGUIBuilder der Firma Segger [8] erstellt und der damit generierte C-Code in die Firmware eingebunden.

Und nun: Ein Blick auf die Messwerte

Die Daten des USB-Sticks sind schnell in Excel importiert und können vielfach ausgewertet werden. So sind zum Beispiel Temperatur-Mittelwerte oder der Verbrauch des Tages mit wenigen Klicks ermittelt. Liniendiagramme zeigen die Abläufe auch grafisch an. In **Bild 8** ist zum Beispiel der interessante Übergang

von Nacht- auf Tagbetrieb auf einer Zeitachse von 4:00 Uhr bis 8:00 Uhr dokumentiert. Um 5:00 Uhr endet die Nachtabsenkung und die Heizung nimmt ihren Tagesbetrieb auf. Damit das Duschwasser rechtzeitig zur Verfügung steht, wird das Warmwasser ab 6:00 Uhr auf über 50 °C aufgeheizt. Danach wird die Tagesheizung fortgesetzt. Die Brenneraktivitäten sind der besseren Übersichtlichkeit halber auf der negativen Achse aufgetragen.

Erfahrungen und erste Resultate

Erst einmal ein Wort zur Stabilität: Der Heizkessel-Monitor ist bei mir seit Anfang 2016 ohne Unterbrechung in Betrieb und das ohne einen einzigen Ausfall. Dies beweist, dass der ESP8266 mittlerweile sehr zuverlässig arbeitet. Und wieviel Brennstoff ist eingespart worden? Das werden die Auswertungen dieses Winters zeigen. Mit wenigen Klicks kann ich den Verbrauch pro Tag ermitteln, mit der Umgebungstemperatur korrelieren und die Optimierungserfolge ableiten.

Der Betrieb ist auf meinen Lebensrhythmus eingestellt, das Warmwasser steht dann zur Verfügung, wenn es erforderlich ist und wird nicht ohne Bedarf mehrfach im Tagesverlauf aufgeheizt. Außerdem hat der Monitor schon gute Dienste bei der Entstörung des Brenners geleistet. So konnte ich dem Heizungsmonteur sagen, wann der Brenner zuletzt aktiv war und wann er hätte wieder anspringen sollen.

Mit diesem Heizkessel-Monitor birgt mein Heizkessel keine Geheimnisse mehr und einer Optimierung seines Betriebes steht nichts mehr im Wege. Nicht zuletzt hat mir die ganze Entwicklung viel Spaß gemacht und meine Erfahrungen vertieft. Das wünsche ich Ihnen auch bei Ihren Projekten! ⏮

(150578)

Weblinks

[1] Elektor-Buch zum ESP8266: www.elektor.de/das-esp8266-praxisbuch

[2] „WLAN kompakt und autonom“, Elektor 4/2016: www.elektormagazine.de/150094

[3] STM32-IDE: www.openstm32.org/HomePage
STM32-Library: http://mikrocontroller.bplaced.net/wordpress/

[4] Sensoren bei AliExpress: https://de.aliexpress.com/item/Free-Shipping-1pcs-DS18B20-Stainless-steel-package-1-meters-waterproof-DS18b20-temperature-probe-temperature-sensor-18B20/32305869288.html

[5] Espressif: http://bbs.espressif.com/

[6] Cherts-Kit: https://github.com/CHERTS/esp8266-devkit
www.esp8266.com/viewtopic.php?f=9&t=820

[7] Software für das Projekt: www.elektormagazine.de/150578

[8] emWinGUIBuiler: www.segger.com/emwin-guibuilder.html

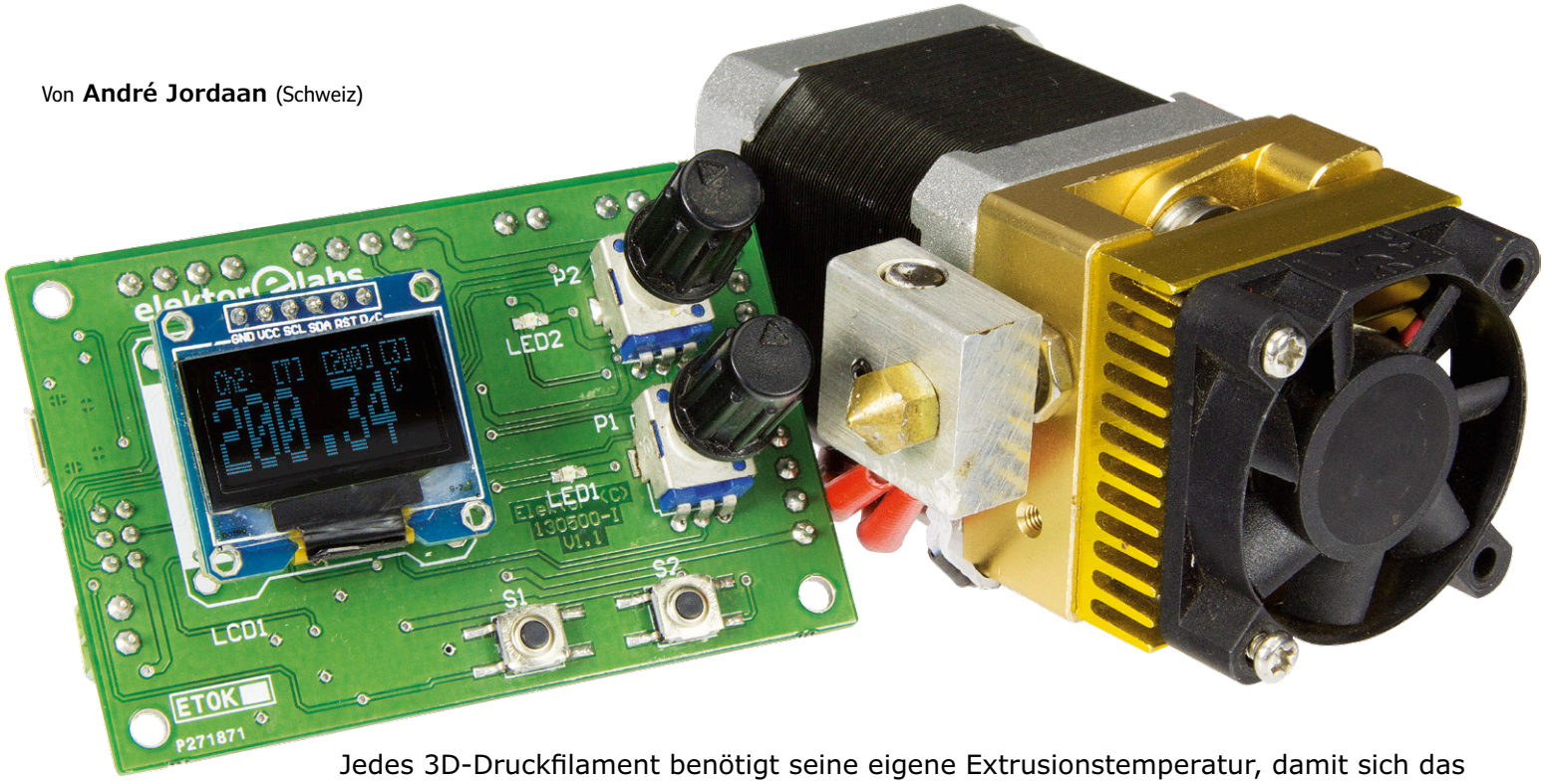
[9] Projekt auf Elektor /Labs: www.elektormagazine.com/labs/monitor-and-data-logger-for-a-heating-boiler

Anzeige

Temperaturregler für 3D-Drucker

Bringt auch Kaninchen warm über den Winter

Von **André Jordaan** (Schweiz)



Jedes 3D-Druckfilament benötigt seine eigene Extrusionstemperatur, damit sich das beste Ergebnis erzielen lässt. Auch wenn für jeden Einsatz das gleiche Material verwendet wird, kann die optimale Temperatur selbst von der Farbe des Filaments abhängen. Deshalb ist ein exakter Temperaturregler für den Extruderkopf kein überflüssiger Luxus. Und wenn wir schon dabei sind, warum nicht auch die Mattentemperatur regeln?

Dieses Projekt entstand, als ich eine X2-Standfräse in eine CNC-gesteuerte Fräse umbaute. Der Weg zu einem 3D-Drucker ist dann nicht mehr weit. Es müssen nur ein Druckkopf und eine Vorrichtung, die die Temperatur des Druckkopfes und der Matte (auf der das Objekt gedruckt wird) genau regelt. Ein Arduino Micro mit OLED-Display, ein paar Thermistoren und ein paar Heizelemente angeschlossen; und schon war der erste Prototyp auf dem Steckbrett entstanden. Als dieser Regler funktionierte, wurde mir klar, dass er auch in anderen Anwendungen eingesetzt werden kann, etwa um einen Kaninchenstall im Winter gemütlich zu temperieren oder empfindliche Außenpflanzen vor Frost zu schützen. Nachdem ich versehentlich etwas so Nützliches erschaffen hatte, schien es eine gute Idee, ein reproduzierbares Projekt daraus zu machen.

Die wichtigsten Eigenschaften

Wie **Bild 1** zeigt, handelt es sich bei dem Projekt um einen Zweikanalregler mit optisch isolierten Ausgängen.

PID-gesteuert

Die beiden Kanäle regeln die Temperaturen von Druckkopf und Matte durch Pulsbreitenmodulationssignale (PWM), die die 12-V/40-W-Heizelemente über MOSFETs ein- und ausschalten. Ein PID-Regelkreis (später mehr dazu) hält die Ist-Temperaturen mit nur einem halben Grad Abweichung von den Soll-Temperaturen konstant. Die Soll-Temperaturen werden einfach durch Potentiometer eingestellt.

Temperatursensoren? Analog, digital, völlig egal!

Zur Messung der Temperatur kann der Regler entweder die populären digitalen DHT-Sensoren (zum Beispiel DHT22, auch AM2302 genannt und im Internet auch als DS18B20 erhältlich) oder analoge Thermistoren (auch bekannt als NTCs) verwenden, sogar pro Kanal unterschiedlich. Thermistoren ertragen hohe Temperaturen, und genau das brauchen wir für einen Extruderkopf. Da die in 3D-Druckerfilamenten verwendeten Materialien einen Schmelzpunkt von etwa 200 ºC auf-

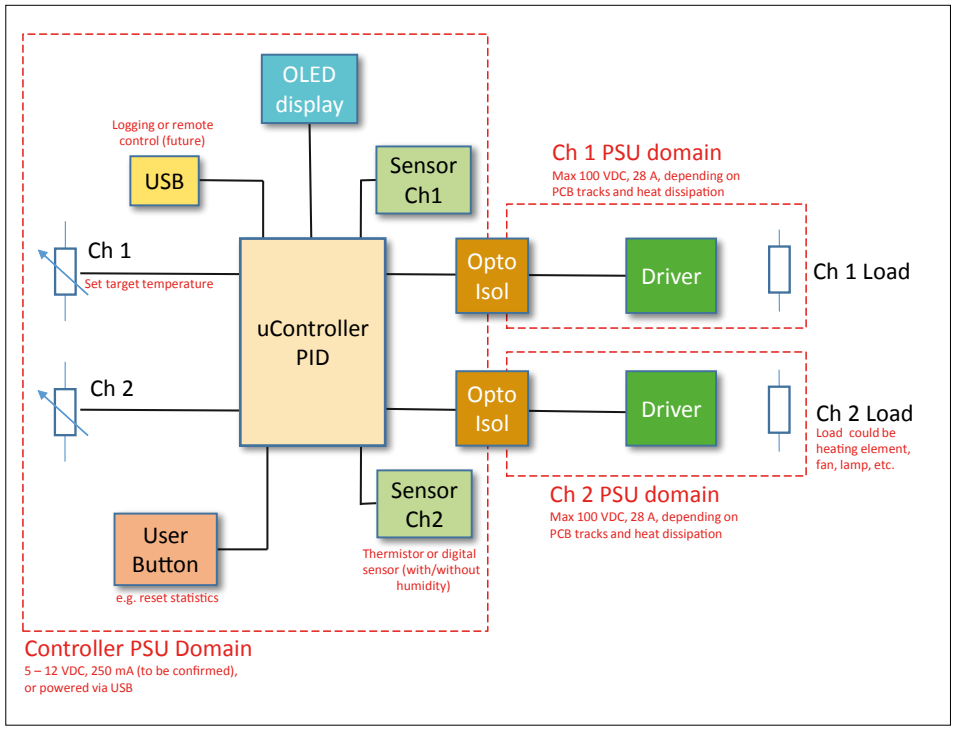


Bild 1. Der Temperaturregler ist mehr als nur ein Controller, der einen Leistungstransistor steuert.

weisen, habe ich einen 100-kΩ-Glaskugel-Thermistor mit einer maximalen Temperatur von 300 °C verwendet. Um das Optimum aus einem Thermistor herauszuholen, wird die Steinhart-Hart-Gleichung verwendet, um die Temperatur zu berechnen.

Datenerfassung

Temperaturen und PID-Parameter werden zur Erfassung über USB zum PC gesendet.

Grafikdisplay

Das SSD1306-kompatible OLED-Display zeigt je nach Software (ja, es gibt zwei Versionen) die tatsächliche Druckkopftemperatur, die Soll-Temperatur und deren Fehler sowie die höchsten und niedrigsten Temperaturwerte seit dem Einschalten (oder seit der letzten Betätigung der User-Taste). Die Anzeige wechselt alle fünf Sekunden zwischen Kanal 1 und Kanal 2. Die rechte obere Ecke zeigt den gerade verwendeten der drei PID-Parametersätze:

- genau - kleinster Fehler, aber langsam
- normal – mittlerer Fehler, akzeptabel reagierend
- aggressiv - größter Fehler, aber schnell

In der oberen linken Ecke des Displays wird ein drehendes Symbol gezeigt, das zeigt, dass der Controller noch läuft. Wenn dies nicht der Fall sein sollte, startet ein Watchdog-Timer das System neu. Bei einem digitalen Temperatursensor wie dem DHT22, der einen relativen Feuchtigkeitssensor beherbergt, wird auch die Feuchte angezeigt.

Die Schaltung

Schauen wir uns die Schaltung in **Bild 2** an. Im Mittelpunkt steht natürlich der Controller IC1, ein 8-bit-AVR-Mikrocontroller ATmega32U4 mit USB-Schnittstelle. Mit einem 16-MHz-Takt wird alles von der Arduino-Programmierumgebung (Arduino Micro oder Leonardo einstellen) unterstützt. Sie müssen lediglich den richtigen Bootloader aus dem Arduino-Environment in den Controller übernehmen. Dazu kann der ICSP-Stecker K2 verwendet werden. Die beiden Potis P1 und P2 stellen die Soll-Temperatur für die beiden Kanäle ein. Die User-Taste S2 setzt je nach Softwareversion die vom Programm gesammelten Werte zurück oder zwingt den Sensor in den digitalen oder analogen Modus.
Digitale Temperatursensoren werden an K7 und K8 angeschlossen, Thermistoren an K3 und K4. Eine Programmversion testet zunächst die Thermistoren, und wenn sie keine aussagekräftigen Werte erhält (sei es, weil die Thermistoren defekt oder gar nicht angeschlossen sind), geht sie zu den digitalen Sensoren über. Die andere Version der Software muss für die Arbeit mit analogen oder digitalen Sensoren explizit konfiguriert werden.
R8 und R9 legen die Ausgänge der Thermistoren auf die Hälfte der Betriebsspannung. Die Widerstände gelten für 100-kΩ-Thermistoren und sollten für andere Typen angepasst werden. Der richtige Thermistor hängt wiederum von der Anwendung ab.
Die beiden Heizkanäle sind mit IC3/T1 und IC4/T2 realisiert. Es handelt sich um optisch isolierte Treiber, so dass Fehler aufgrund von Rauschen und Störungen an den Heizungsversorgungen nicht den Controller stören oder sogar beschädigen können. Damit dies wie vorgesehen funktioniert, ist jegliche Verbindung von V_{IN} zu V_{IN1} oder V_{IN2} auszuschließen. Auch die Massen dürfen keinen Kontakt zueinander haben, und dies kann weniger

Arduino-Bibliotheken

Die Software für dieses Projekt basiert auf freien Bibliotheken, die von Arduino-Enthusiasten kostenlos zur Verfügung gestellt werden. Solche Bibliotheken entwickeln sich weiter, so dass es passieren kann, dass sie schon nicht mehr mit unserer Software harmonieren, wenn Sie diese Zeilen lesen. Zum Glück dürfen wir die „brauchbaren" Versionen weitergeben und in das Download-Paket [1] einpacken. Die neuesten Versionen der Bibliotheken (die möglicherweise nicht von uns getestet wurden) finden Sie auf GitHub:

https://github.com/adafruit/Adafruit_SSD1306
https://github.com/adafruit/Adafruit-GFX-Library
https://github.com/br3ttb/Arduino-PID-Library
https://github.com/PaulStoffregen/Time
https://github.com/PaulStoffregen/TimeAlarms
https://github.com/RobTillaart/Arduino/tree/master/libraries/DHTlib

offensichtlich sein als Sie denken, da der Controller auch über USB (K1) an denselben PC wie der 3D-Drucker oder eine andere Last angeschlossen sein könnte. Die Heizungen und ihre **Gleich**stromversorgung sollten nur an K5 und K6 angeschlossen werden und an sonst nichts. Versuchen Sie nicht, Wechselstromlasten zu schalten! Nach Datenblatt können die Transistoren jeweils 28 A schalten (und vertragen Betriebsspannungen von bis zu 100 V), aber die Platine kann nicht so viel Strom verkraften. Für einen sicheren Betrieb sollte der Laststrom deshalb 2,5 A pro Kanal nicht überschreiten. Die Draufgänger unter Ihnen können ja versuchen, den maximalen Strom durch Kühlung der Transistoren oder/und Verstärkung der Leiterbahnen zu erhöhen.

Übrigens: Ich rede hier ständig über Heizelemente, aber es ist natürlich genau so möglich, eine Heizung und einen Lüfter anzuschließen - oder eine IR-Lampe für den gleichen Zweck.

Das OLED-Grafikdisplay LCD1 ist eine Art Standardmodul und kann leicht im Internet beschafft werden. Es lässt sich entweder im SPI- oder I²C-Modus konfigurieren. Ich habe mich für den SPI-Modus entschieden, auch wenn Schaltbild wie Fotos auf der Displayplatine die typischen I²C-Labels zeigen. Lassen Sie sich davon nicht verwirren!

Die Stromversorgung der Schaltung kann durch ein gewöhnliches Steckernetzteil mit nicht mehr als 12 V Ausgangsspan-

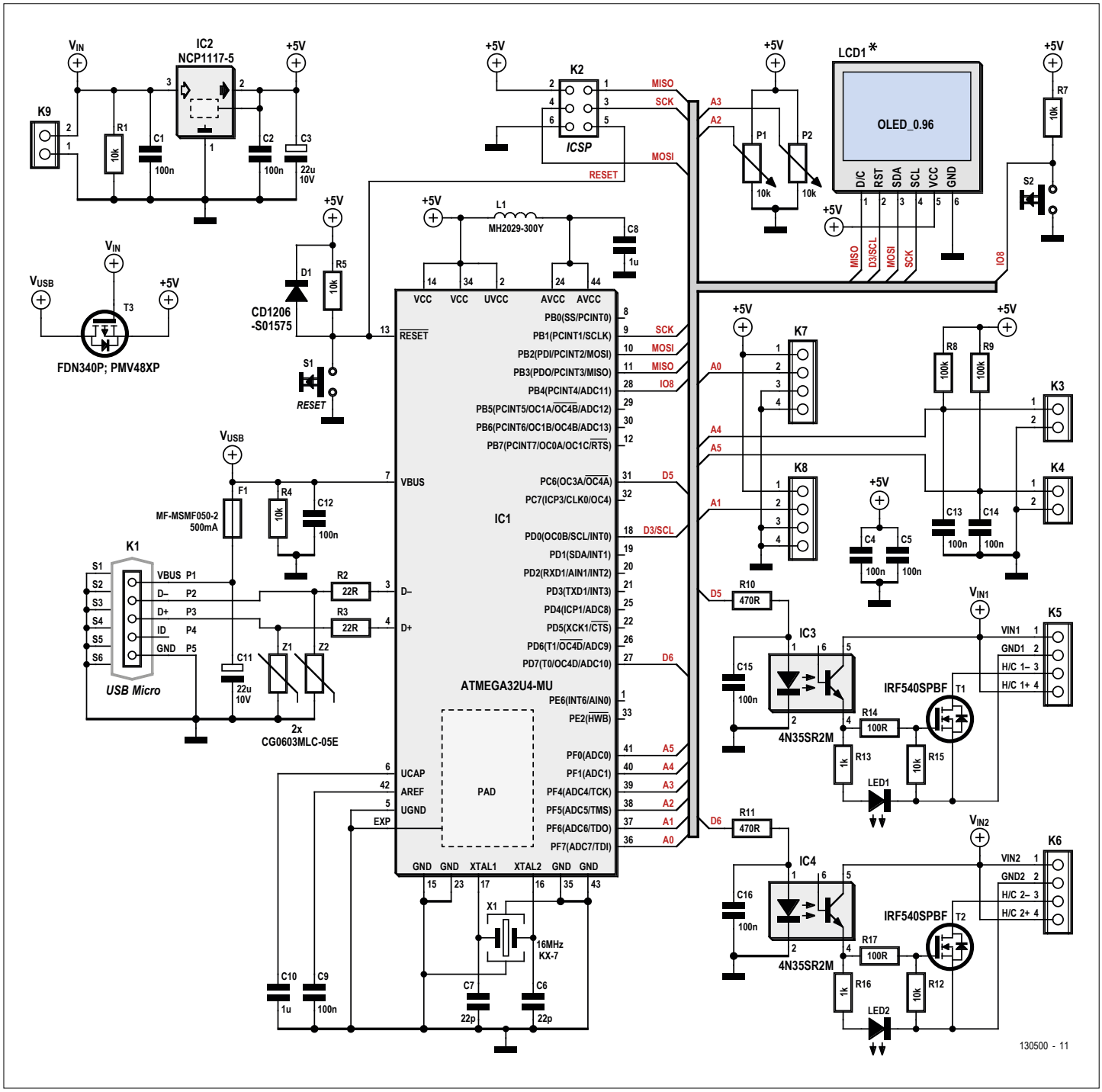


Bild 2. Die Schaltung des 2-Kanal-Temperaturreglers mit den optisch isolierten Ausgängen.

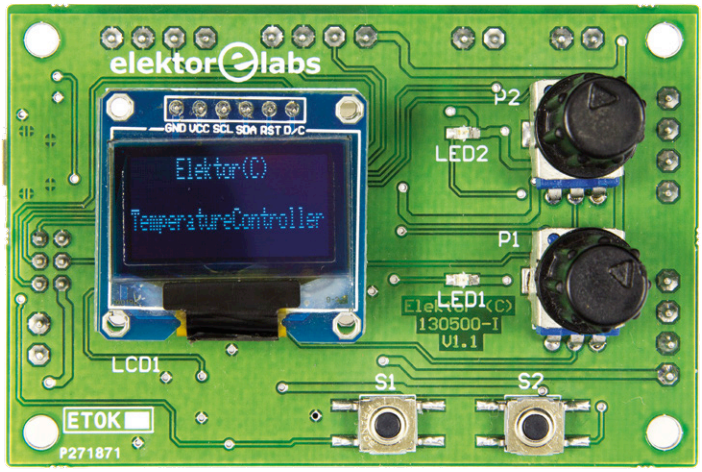


Bild 3. Bedienseite der Temperaturregler-Platine.

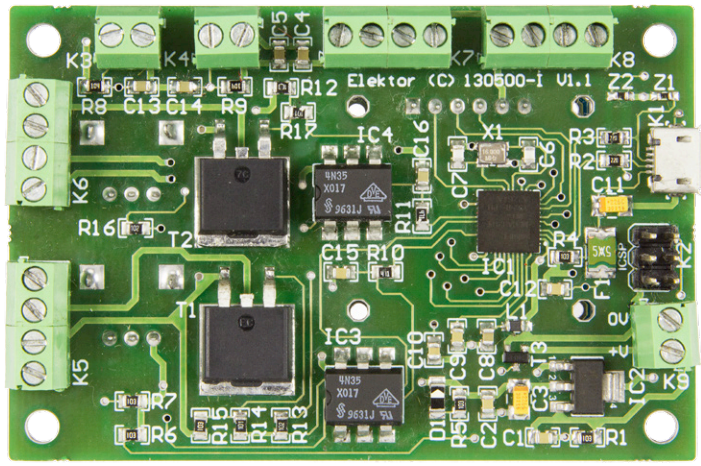


Bild 4. Fast alle Bauteile befinden sich auf dieser Platinenseite.

nung erfolgen. Es muss nicht viel Strom liefern, da die Lasten ohnehin ihre eigenen Stromversorgungen mitbringen. Es ist sogar möglich, den Regler über USB zu versorgen.

Die Software

Obwohl die Aufgaben der Software einfach genug erscheinen (ein paar Sensoren auslesen, zwei Lasten steuern und interessante Daten anzeigen), war eine beträchtliche Arbeit erforderlich, um die Software zu erstellen. Zum Glück war eine Menge von dem, was für das Programm erforderlich war, im Internet zu finden (siehe **Kasten** *Arduino-Bibliotheken*), so dass es im Wesentlichen darauf ankam, herauszufinden, wie man all diesen Code verwendet. Ich musste nur etwas Code schreiben, um die Bibliotheken zusammenzukleben. Am Ende entstanden zwei Softwareversionen:

- der von Elektor „eingefrorene" Code [1];
- der Code, der weiterentwickelt werden kann [2].

Grundsätzlich haben beide Versionen die gleichen Ziele, aber sie unterscheiden sich in Details. Wenn Sie also eine Version auswählen und feststellen, dass sie nicht dem entspricht, was hier beschrieben und versprochen wird, dann nehmen Sie bitte die andere Version und versuchen Sie es erneut.

Digitalsensoren

Die Bibliothek für die digitalen Sensoren heißt „dht". Da dies beliebte Bauteile sind, gibt es gleich mehrere Bibliotheken, die den gleichen Namen tragen. Achten Sie darauf, dass Sie die richtige erwischen (im Download [1]).

Steinhart-Hart

Sie haben sich sicher schon gefragt, was es mit dieser geheimnisvollen Gleichung auf sich hat. Die analogen Sensoren sind nun mal analog und so liest der A/D-Wandler des Controllers eine Spannung, bevor er sie in einen Widerstandswert umrechnet. Die Beziehung zwischen Temperatur- und Thermistorwiderstand ist aber nicht linear und kann durch das (erweiterte) Steinhart-Hart-Polynom angenähert werden:

$$1 / T = A_0 + A_1 \times \ln(R) + A_2 \times [\ln(R)]^2 + A_3 \times [\ln(R)]^3$$

Solch eine Gleichung zu verwenden, hat nur dann Sinn, wenn Sie für R8 und R9 auch Widerstände mit 1 % Fehler verwenden und die Schaltung nicht mit der ungenauen USB-Spannung versorgen. Es gibt auch einen ganz pragmatischen Grund für diese Gleichung, nämlich, keine große Lookup-Tabelle im Programmspeicher unterbringen zu müssen. Das ist bei einem solch umfangreichen Programm (das später ja auch noch durch zusätzliche Features erweitert werden kann) nicht ohne Bedeutung.

Die Koeffizienten A_0...A_3 hängen vom verwendeten Thermistor ab und können mit einem kleinen Tool [3] (in C oder Java, ich habe die C-Version verwendet) ermittelt werden. Dieses Tool „coeff.exe" verwendet die Temperatur-Widerstands-Tabelle aus dem Datenblatt des Thermistors, die Sie zuvor als „simu.txt" im selben Ordner wie „coeff.exe" gespeichert haben (die alte Datei überschreiben), und berechnet anschließend das Polynom. Es gibt die vier Koeffizienten aus (und einige andere Kontrolldaten). Ich hatte ursprünglich die Version 0.1 des Tools. Sie arbeitete perfekt, aber mit der aktuellen Version 1.0 gab es bei mir Schwierigkeiten.

Noch einmal: Der PID-Controller

Nach der Messung wird die Temperatur dem PID-Regler zugeführt. Es gibt Leute, für die PID-Regler ein rotes Tuch sind (weil sie nicht wissen, wie man sie implementieren kann), aber dafür gibt es eigentlich überhaupt keinen Grund, kann man doch einfach eine Bibliothek aus dem Internet ziehen und diese Arbeit erledigen lassen. Sie müssen sich nicht schämen, ich habe es genauso gemacht. Noch besser: Zu der Bibliothek, die ich verwende, gibt es eine großartige Dokumentation [4], die nicht nur erklärt, was ein PID-Regler ist, sondern auch, wie man ihn benutzt.

Kurz gesagt: Ein PID-Regler verwendet drei Werte, die aus den Ist-Werten eines Parameters abgeleitet werden, um ein Regelsignal zu erzeugen, das den Ist-Wert näher an den Soll-Wert des Parameters verschiebt. Diese drei Werte sind: der Ist-Wert (proportional oder *P*), der akkumulierte Fehler zwischen Ist- und Soll-Wert (Integral oder *I*) und die Änderungsrate des Ist-Wertes (Differential oder *D*). Der PID-Regler mischt die Anteile der drei Werte, um ein Steuersignal für den Ausgangswert zu erzeugen. Die Art und Weise, wie der Ausgang reagiert, hängt davon ab, wie *P*, *I* und *D* gemischt (oder gewichtet) werden. Ein starker Anteil von

P führt zu einer schnellen Reaktion, aber der Ausgangswert wird überreagieren, was zu starken Über- und Unterschreitungen des Soll-Werts und sehr wahrscheinlich zur Stabilisierung auf einen falschen Wert führen wird. Durch Hinzufügen von *I* erreicht man am Ende zwar den Soll-Wert (akkumulierter Fehler = 0), aber das Signalschwingen wird verstärkt. Werfen wir einiges *D* in den Ring, wird das Signalschwingen gedämpft, aber auf Kosten der Reaktionsgeschwindigkeit des Systems. Die Kunst ist es, die Anteile der drei Ingredienzien so zu wählen, dass sich das System bestmöglich verhält. Dafür gibt es keine Universallösung, es hängt vom System ab und von dem, was man von ihm erwartet. Es gibt verschiedene Techniken, um einen PID-Regler abzustimmen, von mathematisch bis empirisch, und viel Papier ist zu diesem Thema bedruckt worden. Hier ist sicher nicht der Raum, um dies alles zu erklären, und eigentlich bin ich auch kein Experte für PID-Regler, sondern ich arbeite mit der „Trial and error"-Methode. Bitte durchsuchen Sie das Internet, wenn Sie es besser machen wollen!

Die drei erwähnten PID-Einstellungen aggressiv, normal und genau bieten unterschiedliche Voreinstellungen für die P-, I- und D-Werte. Der Rest des Programms beschäftigt sich vor allem mit der Darstellung der Ist- und Soll-Werte auf dem OLED-Display. Ein Grafikdisplay erfordert viel Programmierarbeit. Ein Glück, dass sich viele Leute vor mir damit beschäftigt und Arduino-Bibliotheken dafür veröffentlicht haben. Ein großes Dankeschön an die Open-Source-Bewegung, denn so ist mein Leben viel einfacher!

Stückliste

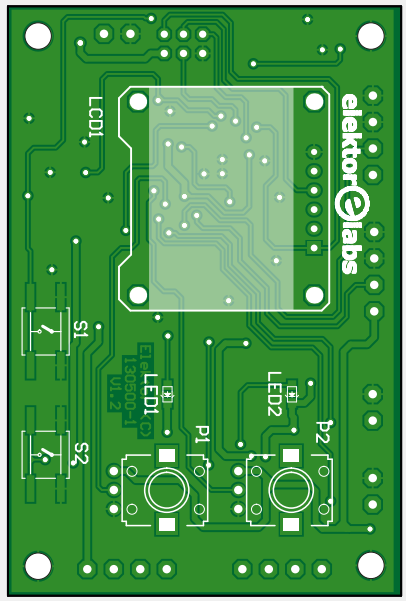
Widerstände:
alle SMD 0805, 0,1 W, 1 %
R1,R4,R5,R7,R12,R15 = 10 k
R2,R3 = 22 Ω
R8,R9 = 100 k (siehe Text)
R10,R11 = 470 Ω
R13,R16 = 1 k
R14,R17 = 100 Ω
P1,P2 = Potentiometer 10 k, linear

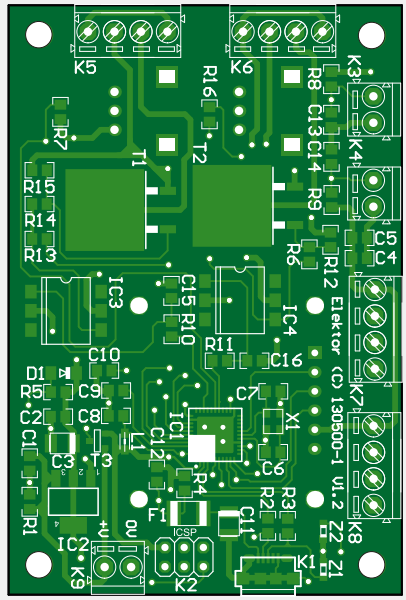
Kondensatoren:
alle SMD 0805
C1,C2,C4,C5,C9,C12,C13,C14,C15,C16 = 100 n
C3,C11 = 22 µ, 10 V, Gehäuse A
C6,C7 = 22 p
C8,C10 = 1 µ

Induktivität:
SMD 0805
L1 = Ferrit MH2029-300Y, 0,025 Ω, 3 A

Halbleiter:
D1 = CD1206-S01575
IC1 = ATmega32U4-MU
IC2 = NCP1117ST50T3G
IC3, IC4 = 4N35SR2M
LED1, LED2 = LED, grün
T1, T2 = IRF540SPBF
T3 = FDN340P

Außerdem:
LCD1 = SPI/I²C-OLED-Display, 0,96", SSD1306-kompatibel
Z1, Z2 = Varistor CG0603MLC-05E
F1 = MF-MSMF050-2, 15 V, 0,5 A, 1812
X1 = Quarz 16 MHz, 2,5x2 mm
S1, S2 = Taster, 6x6 mm, Pins nach außen (gullwing) (z.B. C&K Components KSC321GLFS)
K1 = Micro-USB Type B (z.B. Amphenol FCI 10104110-0001LF)
K2 = 2x3-polige Stiftreihe, Raster 0,1'', vertikal
K3,K4,K9 = 2-polige Platinenklemme, Raster 3,5 mm
K5,K6,K7,K8 = 4-polige Platinenklemme, Raster 3,5 mm
Für LCD1: je 1x6-polige Stift- und Buchsenreihe, Raster 0,1''
Platine Elektor 130500-1





Ein Zwischenfazit

Dieser Artikel zeigt, wie man einen PID-Regler konstruiert, um die Temperatur eines 3D-Druckkopfes genau einzustellen. Er ist jedoch nicht auf diese Anwendung beschränkt, sondern man kann auch noch ganz andere Dinge damit tun. Wegen seiner Vielseitigkeit versuche ich derzeit, den Controller über ein Netzwerk zu steuern, indem ich ihm einen Raspberry Pi als Webserver zur Seite stelle, der mit dem Regler über einen isolierten I²C-Bus kommuniziert (mehr darüber in dem Artikel „Universeller I²C-Bus-Isolator und Pegelwandler" in der Elektor-Ausgabe 7-8/2016 [6]). Ich habe mit dem Schreiben der Software begonnen, aber sie ist noch lange nicht fertig. Informieren Sie sich unter [2] regelmäßig über die Fortschritte! ◀

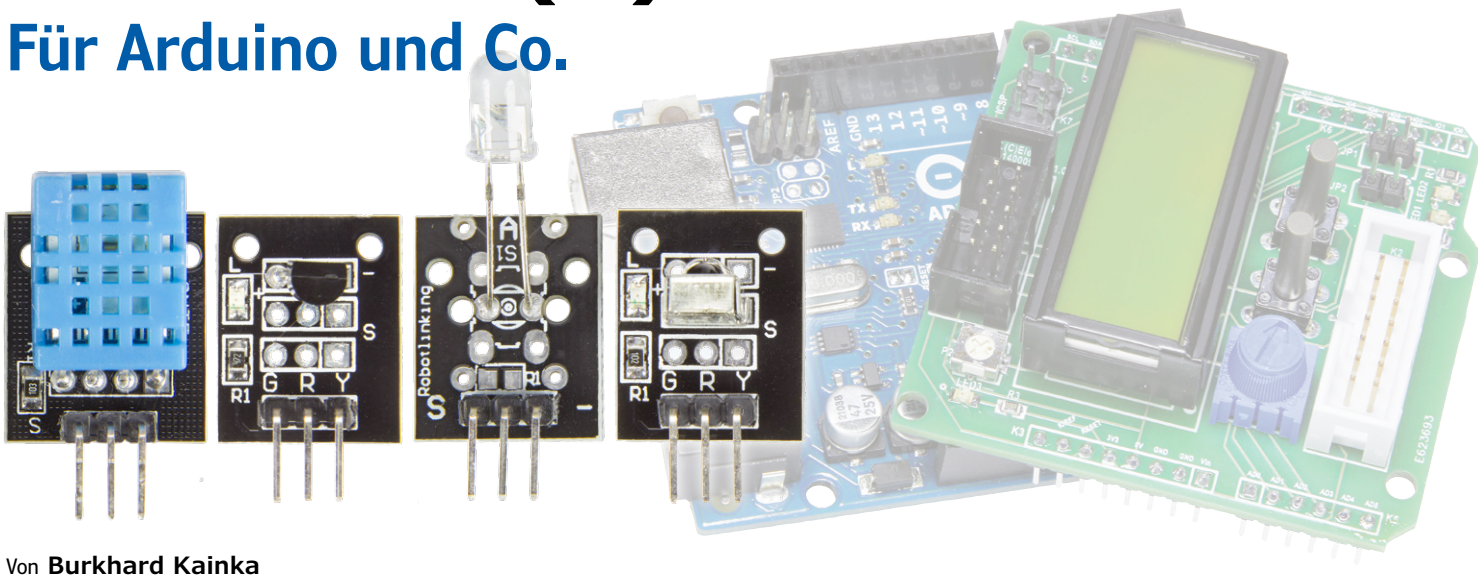
(130500)

Weblinks

[1] www.elektormagazine.de/130500
[2] www.elektor-labs.com/3d-printer-head-and-mat-temperature-controller-using-arduino-130500-i
[3] http://thermistor.sourceforge.net/
[4] http://brettbeauregard.com/blog/2011/04/improving-the-beginners-pid-introduction/
[5] www.elektormagazine.com/labs/universal-i2c-bus-isolator-and-level-adapter-150089
[6] www.elektormagazine.de/150089

Sensoren (3)

Für Arduino und Co.



Von **Burkhard Kainka**

Elektrische Kommunikation lief schon seit den Anfängen der Telegrafie über einen einzelnen Draht. Auch der Eindraht-Bus und ähnliche Protokolle kommen mit nur einer Leitung aus, eine zusätzliche Taktleitung ist nicht nötig. IR-Fernbedienungen funktionieren nach einem ähnlichen Prinzip. In dieser Folge werden wir uns unter anderem mit Temperatur- und Feuchtesensoren sowie dem IR-Sender und -Empfänger aus dem 35-Sensor-Kit beschäftigen.

Wir beginnen mit dem Temperatursensor „18B20 Temp“ aus dem Kit, das im Elektor-Shop erhältlich ist [1]. Der dort verbaute DS18B20 sieht aus wie ein gewöhnlicher Transistor im TO92-Gehäuse, ist aber ein komplexes IC mit einem Temperatursensor und einer besonderen Schnittstelle. Der 1-Wire-Bus wurde von der Firma Dallas entwickelt. Ein oder mehrere Sensoren können an einem einzelnen Draht betrieben werden, wenn man die GND-Leitung nicht mitzählt. Über diesen Draht läuft nicht nur der Datenstrom, das IC kann auch darüber mit Strom versorgt werden. Meist verwendet man aber den dritten Anschluss VDD des DS18B20 für die Stromversorgung und hat dann insgesamt drei Drähte. Auf der Platine des Sensors befindet sich eine zusätzliche LED mit einem Vorwiderstand an der Datenleitung.

Arduino-Software für den 18B20

In die Arduino-IDE müssen zwei Bibliotheken geladen werden: OneWire und DallasTemperature. Beide finden sich auf der CD zum Sensor-Kit und müssen in das Verzeichnis „libraries“ im Arduino Sketchbook-Ordner kopiert werden. Man findet jeweils eine Header-Datei *.h und eine C++-Datei *.cpp. Es kann ganz interessant sein, sich diese Dateien genauer anzusehen. Aber wenn man dabei zu dem Schluss kommt, dass man überhaupt nicht mehr durchblickt, ist das auch nicht schlimm. Man muss ja nur die Header-Dateien in das eigene Programm einbinden und braucht dann nur noch einige wenige Funktionen aufzurufen. Dank Arduino ist alles ganz einfach geworden. Das Programm in **Listing 1** zeigt eine einfache Temperaturabfrage, die sich nach jeweils 500 ms wiederholt. Die Datenleitung wurde

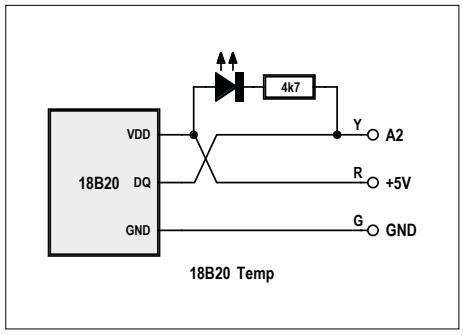


Bild 1. Anschluss des 18B20-Temperatursensors.

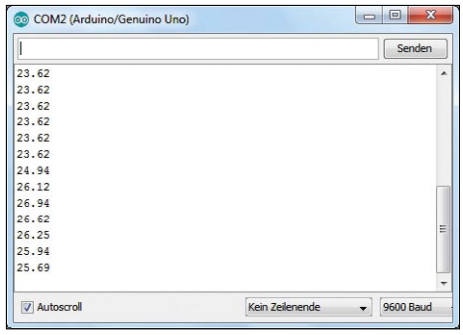


Bild 2. Ausgabe der Temperatur im seriellen Monitor.

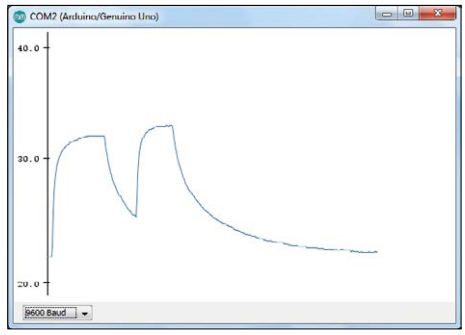


Bild 3. Temperaturverläufe.

Listing 1. Temperaturmessung mit dem DS18B20.

```
#include <OneWire.h>
#include <DallasTemperature.h>
#include <LiquidCrystal.h>
#define ONE_WIRE_BUS A2
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

float temp;
int minTemp;
int maxTemp;
LiquidCrystal lcd(2,3,4,5,6,7);

void setup(void)
{
  Serial.begin(9600);
  sensors.begin();
  lcd.begin(16, 2);
  minTemp = 100;
  maxTemp = -100;
}

void loop(void)
{
  sensors.requestTemperatures();
  temp = sensors.getTempCByIndex(0);
  Serial.println(temp);
  lcd.setCursor(0, 0);
  lcd.print(temp);
  lcd.print (" C   ");
  if (temp < minTemp) minTemp = temp;
  if (temp > maxTemp) maxTemp = temp;
  lcd.setCursor(0, 1);
  lcd.print(minTemp);
  lcd.setCursor(5, 1);
  lcd.print(maxTemp);
  delay (500);
}
```

dabei an A2 (= PC2) gelegt (**Bild 1**). Der gesamte Code zu diesem Artikel kann natürlich wieder bei Elektor heruntergeladen werden [4].
Das Thermometer hat nicht nur eine serielle Ausgabe, sondern zeigt die Temperatur auch auf dem LC-Display, falls man das Elektor Extension-Shield [2][5] verwendet. Und weil in der zweiten Zeile noch Platz ist, haben wir gleich noch ein Minimum/Maximum-Thermometer programmiert.
Die Temperatur wird als Real-Zahl mit zwei Nachkommastellen ausgegeben. Die tatsächliche Auflösung beträgt 0,06 Grad pro Schritt. Die absolute Genauigkeit wird mit 0,5 Grad angegeben. Im seriellen Monitor (**Bild 2**) kann man die Daten sehen. Eine Änderung der Temperatur erreicht man mit dem Berühren des Sensors.
Der serielle Plotter zeigt den zeitlichen Verlauf der Temperatur (**Bild 3**). Bei dieser Messung wurde der Sensor zweimal mit dem Finger erwärmt. Man sieht deutlich die unterschiedlichen Zeitkonstanten beim Erwärmen und Abkühlen. Interessant ist auch, dass der Finger bei der zweiten Berührung offenbar eine höhere Temperatur hatte. Irgendetwas muss in der Zwischenzeit zu einer Temperaturerhöhung geführt haben. Dank der hohen Auflösung des Sensors können auch kleine Unterschiede erkannt werden.

18B20 in Bascom

Bascom unterstützt den Eindraht-Bus, es gibt aber noch keine vollständige Lösung für den 18B20. Also muss man etwas tiefer (ins Datenblatt) eintauchen und einiges selbst programmieren. Das ist zwar mit Arbeit verbunden, aber es eröffnet auch Chancen. Man kann nämlich auch die speziellen Eigenschaften des Sensors nutzen, also zum Beispiel die Auflösung verändern oder die in jedem IC enthaltene und einmalige ID-Nummer auslesen.

Das Bascom-Beispiel (**Listing 2**) zeigt, wie man die Temperatur ausliest. Es müssen zwei Kommandos (Skip ROM, &HCC und Convert T, &H44) gesendet werden. Nach einer Wartezeit für die eigentliche Messung sendet man das Kommando Read Scratchpad (&HBE) und liest dann zwei Bytes aus. Daraus wird eine 16-bit-Zahl und die Temperatur in Schritten von 0,0625 Grad berechnet. War doch gar nicht so schlimm, das Programm ist nur unwesentlich länger als die Arduino-Version. Und auch hier hat man zugleich ein Minimum/Maximum-Thermometer mit LCD-Anzeige.

Das Eindraht-Protokoll

Alles läuft beim Eindraht-Bus über die eine Leitung DQ. Dazu kommen allerdings noch GND und VCC. Die Datenleitung DQ wird im Ruhezustand mit einem Pullup hochgezogen. Der Master kann nun einen Reset-Impuls senden und damit eine Kommunikation beginnen, um dann Kommandos zu senden oder Daten zu empfangen. Beide Partner können Daten auf den Bus legen. Ein 0-Bit wird durch einen 15 µs langen Low-Impuls und einen nachfolgenden, 45 µs langen High-Zustand dargestellt. Ein 1-Bit wird durch einen 60 µs langen Low-Impuls symbolisiert. Zwischen den einzelnen Bits gibt es einen Ruhezustand, in dem die Datenleitung vom Pullup hochgezogen wird.
Der Master (Controller) schickt jeweils einen Reset und dann ein oder mehrere Kommandos. Ein Sensor-Chip antwortet dann mit den gewünschten Daten. Wer sich das Datenblatt des Chips genauer ansieht, findet darin nicht nur das allgemeine Bus-Protokoll sondern auch zahlreiche Kommandos und den Aufbau der zurückgelesenen Datenstruktur. Die Sache wird noch komplexer, weil sogar mehrere Slaves am gleichen Bus liegen können. Man ahnt, wieviel Arbeit es sein muss, das alles selbst zu programmieren. Aber man muss das Rad ja nicht jedesmal neu erfinden, sondern kann fast immer auf fertigen Code zurückgreifen.

Listing 2. Temperaturmessung in Bascom (Auszug).

```
'DS18B20LCD  AD2, PORTC.2
...
Do
  1wreset
  1wwrite &HCC
  1wwrite &H44
  Waitms 800
  1wreset
  1wwrite &HCC
  1wwrite &HBE
  Dat(1) = 1wread(2)
  1wreset
  Temp = 256 * Dat(2)
  Temp = Temp + Dat(1)
  Temp = Temp * 0.0625
  Print Temp
  Tempint = Round(temp)
  If Tempint > Maxtemp Then Maxtemp = Tempint
  If Tempint < Mintemp Then Mintemp = Tempint
  Locate 1 , 1
  Lcd Temp ; " C   "
  Locate 2 , 1
  Lcd Mintemp
  Locate 2 , 5
  Lcd Maxtemp
  Waitms 200
Loop

End
```

Temperatur und Luftfeuchte mit dem DHT11

Der integrierte Feuchte- und Temperatursensor DHT11 sieht auf den ersten Blick aus wie ein einfacher resistiver Feuchtesensor. Dieser Eindruck wird noch verstärkt, wenn man sieht, dass ein analoger Pin vorgeschlagen wird (**Bild 4**). Aber tatsächlich verbirgt sich hinter dem unscheinbaren blauen Gehäuse ein komplexer Sensor mit einer digitalen Schnittstelle.

Die chinesische Firma Aosong hat sich hier etwas einfallen lassen, das nur auf den ersten Blick an den Eindraht-Bus von Dallas erinnert. Jede Messung wird durch einen mindestens 18 ms langen Low-Impuls vom Master initiiert. Danach müssen insgesamt 40 Bits ausgelesen werden, was der Master jedes Mal mit einem 80 µs langen Low-Impuls anfordert. Der Sensor antwortet mit High-Impulsen, wobei eine Länge von maximal 28 µs für eine Null steht und eine Länge von 70 µs für eine Eins. Die 40 Bits enthalten dann jeweils ein Higbyte und ein Lowbyte für die Feuchte und die Temperatur und ein zusätzliches Parity-Byte zur Überprüfung der korrekten Übertragung. Beim DHT11 stehen die Lowbytes jeweils auf Null, sodass keine Nachkommastellen übertragen werden. Es gibt aber auch noch den DHT22 mit dem gleichen Protokoll, der auch die Nachkommastellen liefert. Deshalb können mit der gleichen Library beide Typen ausgelesen werden.

Und wieder ist man froh, dass jemand sich bereits die Mühe gemacht hat, das komplizierte Protokoll in eine Arduino-Library zu gießen. Die Verwendung ist ganz einfach, nachdem man das Library-Verzeichnis DHT von der CD in den Arduino Library-Ordner kopiert hat. Das Beispiel (**Listing 3**) zeigt, wie man beide Messwerte auslesen und auf dem LCD darstellen kann. Als Anschluss für die Datenleitung wurde wieder der Pin AD2 gewählt.

Im Datenblatt wird die absolute Genauigkeit der Temperaturmessung ganz bescheiden mit 2 Grad angegeben, die der Luft-

Listing 3. DHT11 in Arduino-C.

```
//DHT11LCD, pin AD2

#include <dht.h>
#define dht_apin A2
#include <LiquidCrystal.h>
LiquidCrystal lcd(2,3,4,5,6,7);
float temperature;
float humidity;

dht DHT;

void setup(){
  Serial.begin(9600);
  delay(500);
  delay(1000);
  lcd.begin(16, 2);
}

void loop(){
   DHT.read11(dht_apin);
   humidity = DHT.humidity;
   temperature = DHT.temperature;
   Serial.print("Humidity = ");
   Serial.print(DHT.humidity);
   Serial.println(" %  ");
   Serial.print("Temperature = ");
   Serial.print(temperature);
   Serial.println(" C  ");
   lcd.setCursor(0, 0);
   lcd.print(temperature);
   lcd.print (" C   ");
   lcd.setCursor(0, 1);
   lcd.print(humidity);
   lcd.print (" %   ");
   delay(2000);
}
```

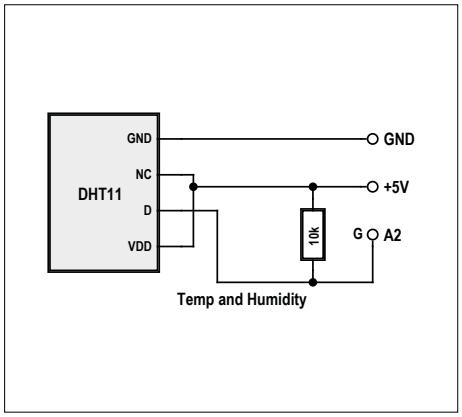


Bild 4. Anschluss des Feuchte-/Temperatursensors DHT11.

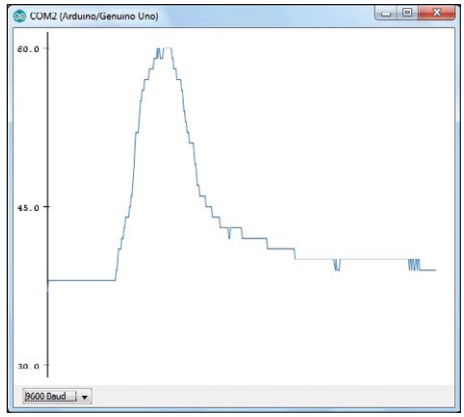


Bild 5. Messung der Luftfeuchte bei Berührung mit dem Finger.

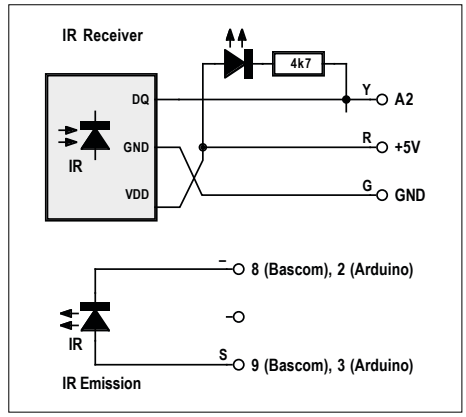


Bild 6. IR-Empfänger und -Sender.

feuchte mit 5 %. Die Luftfeuchte ist schwer zu beurteilen, weil alle üblichen Feuchtemesser auch nur grobe Schätzeisen sind. Aber bei der Temperatur sah es zumindest beim gemessenen Exemplar recht gut aus. Ein vorhandenes digitales Thermometer zeigte 23,8 °C, der DHT11 lieferte 24 °C, und der DS18B20 meldete 23,37 °C, wobei man jedes Mal lange warten muss, nachdem man einen Sensor berührt hat.

DHT11 und Bascom

In Bascom gibt es zwar viele vorbereitete Befehle und Funktionen, aber der DHT11 wird nicht direkt unterstützt. Bei den meisten Themen ist es aber durchaus erfolgversprechend, einmal im Netz zu recherchieren, ob schon mal jemand damit gearbeitet hat. In diesem Fall führte die Suche zu einem Ergebnis im Bascom-Forum. Ein User namens Grütze hat ein Programm DHT11LCD.bas geschrieben, das genau das tut, was es soll. Der Code ist gut lesbar und zeigt im Grunde genau das, was im Datenblatt steht. Es waren nur geringe Anpassungen nötig, um das Extension-Shield zu verwenden und den Sensor am Pin AD2 laufen zu lassen. Außerdem wurde eine serielle Ausgabe mit eingebaut, und zwar zunächst nur für die Feuchtedaten, weil beabsichtigt war, sie mit dem seriellen Plotter aus der Arduino-IDE darzustellen (**Bild 5**).
Die entscheidenden Dinge passieren in der Function Get_dht11(). Hier wird genau das ausgeführt, was für den Arduino in der entsprechenden Bibliothek steht. Und so kann man auch mit Bascom eine vollständige kleine Wetterstation realisieren (**Listing 4**).

Infrarot-Fernbedienung

Infrarot-Fernbedienungen kennt man von Fernsehern und andern Geräten der Unterhaltungselektronik. Es gibt sehr viele unterschiedliche Hersteller und Protokolle, die nicht zueinander kompatibel sind. Eine Fernbedienung muss deshalb genau zum Gerät passen. Allen Typen gemeinsam ist, dass das Signal einer Infrarot-Sendediode mit einer Frequenz zwischen 30 kHz und 40 kHz moduliert wird. Dieses Signal wird dann nach unterschiedlichen Mustern geschaltet, indem einzelne Impulspakete abgesandt werden. Ein integrierter Infrarotempfänger empfängt die Signale mit einer Fotodiode, verstärkt und filtert sie und demoduliert sie zu einem digitalen Signal. Die internen Filter sind zwar für eine bestimmte Frequenz im Bereich 30 bis 40 kHz dimensioniert, aber die Bandbreite reicht aus, um auch andere Fernbedienungen in geringerer Entfernung zu empfangen.
Eine IR-Fernbedienung kann man auch für allgemeine Fernsteuer- oder Schaltaufgaben verwenden. Das Sensor-Kit enthält eine IR-Sendediode (IR Emission) und einen integrierten IR-Empfänger (IR Receiver, **Bild 6**). Zusammen mit einem Arduino kann man damit wahlweise IR-Signale empfangen oder IR-Signale aussenden. Hier soll ein Programm vorgestellt werden, das beides kann. Über zwei Tasten werden Kommandos gesendet, die im Empfänger ausgewertet werden, um einen Ausgang zu schalten. Dieselbe Aufgabe soll einmal in Bascom und einmal mit Arduino-C gelöst werden. Und beide Programme sollen soweit kompatibel sein, dass der Bascom-Controller dem Arduino-programmierten Uno sagen kann, was zu schalten ist und umgekehrt.

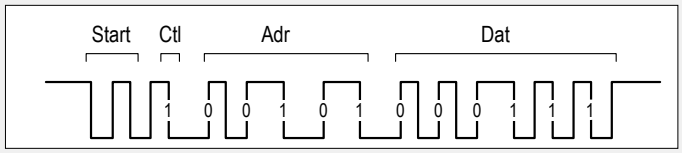
Ein häufig verwendeter Standard für IR-Fernbedienungen ist RC5 von Philips (siehe Kasten). Dafür gibt es einfache Befehle in Bascom, sodass die Dekodierung nicht schwer fällt. **Listing 5** zeigt einen einfachen RC5-Empfänger und -Sender in Bascom. Alle empfangenen Daten werden auf dem LCD dargestellt. Das Kommando wird zusätzlich über die serielle Schnittstelle gesen-

Listing 4. Verwendung des DHT11 in Bascom (Auszug).

```
'DHT11LCD an AD2, PORTC.2
...
Do
   If Get_dht11() = 1 Then
      Print Humidity
      Locate 1 , 1
      Lcd "H: " ; Humidity ; " %  "
      Locate 2 , 1
      Lcd "T: " ; Temperature ; " C  "
    End If
    Waitms 1000
Loop

End
```

Das RC5-Protokoll

Infrarot-Fernbedienungen für Fernsehgeräte, Videorecorder und andere Geräte der Unterhaltungselektronik arbeiten zum Teil nach dem von Philips definierten RC5-Standard. Man verwendet dabei modulierte Lichtsignale im Bereich 30 kHz bis etwa 40 kHz. Die Fernbedienung sendet einzelne Bursts aus, also Impulspakete mit Längen von 0,888 ms oder 1,776 ms. Bei einer Modulationsfrequenz von 36 kHz enthält ein kurzer Burst 32 Einzelimpulse, ein langer 64. Das gesamte Datenpaket dauert etwa 25 ms und wird alle 100 ms wiederholt, solange eine Taste gedrückt wird.



Das Protokoll verwendet ein Bi-Phasen-Signal. Ein Bit hat eine Länge von 1,776 ms. Wenn der 36-kHz-Puls in der ersten Hälfte dieser Zeit liegt, bedeutet das eine logische Null, eine logische Eins wird durch einen Impuls in der zweiten Hälfte dargestellt. Das Signal wird mit einer immer gleich bleibenden Startsequenz eingeleitet. Dann folgen drei Datenbereiche.

- Das Control-Bit (Ctl) wechselt bei jeder Tastenbetätigung zwischen 0 und 1. Der Empfänger kann auf diese Weise unterscheiden, ob eine Taste einmal lange oder mehrmals kurz gedrückt wird.
- Die Geräte-Adresse (Adr) enthält 5 Bits, wobei höherwertige Bits zuerst übertragen werden. Übliche Geräteadressen sind 0 für Fernsehgeräte und 5 für Videorecorder. Auf diese Weise können mehrere Fernbedienungen im selben Raum eingesetzt werden.
- Der Datenbereich (Dat) enthält 6 Bits für bis zu 64 unterschiedliche Tasten. Die Nummerntasten 0 bis 9 erzeugen Codes von 0 bis 9. Auch hier werden die höchstwertigen Bits zuerst gesendet.

det. Außerdem wird es ausgewertet, um einen Ausgang am Port B zu schalten. An B.2 liegt die LED2 auf dem Extension-Shield. Daher haben wir programmiert, dass man sie mit der Fernbedienungs-Taste 2 einschalten und mit der 0 ausschalten kann. Der RC5-Empfänger kann an jeden beliebigen Eingangspin angeschlossen werden. Deshalb wurde auch hier wieder der Eingang PC3 (AD3) eingesetzt. Bascom verwendet für den Befehl Getrc5 im Hintergrund den Timer0-Interrupt, den man global freigeben muss. Zusätzlich wird für den Eingang PC3 der interne Pullup eingeschaltet. Dann ist es nämlich möglich, die Steuerung auch ohne angeschlossenen IR-Empfänger allein zum Senden zu verwenden. Bei einem hochohmigen Eingang ohne Pullup könnte der offene Eingang dagegen einen Low-Zustand einnehmen und das Programm hängen lassen.

Als Sendeausgang (Befehl Sendrc5 in Bascom) wird grundsätzlich PB1 (Arduino Pin 9) verwendet, weil der Ausgang OC1A des Timer 1 an diesem Pin liegt, der für die Erzeugung des 36-kHz-Sendesignals genutzt wird (Achtung, in der Arduino-C++-Software wird ein anderer Ausgang für diese Aufgabe verwendet, sodass man diesmal nicht die gleiche Anschlussbelegung verwenden kann).
Der Ruhezustand dieses Portpins muss in Bascom vorab durch einen Portbefehl festgelegt werden und soll hier in den Pausen low sein. Für jedes Impulspaket schaltet die Software dann vom Port auf den Timerausgang um. Weil die IR-Diode keinen Vorwiderstand besitzt, ist es sinnvoll, sie gegen einen anderen Ausgangsport zu schalten, um den zweifachen Innenwiderstand zur Strombegrenzung zu verwenden. Hier wurde PB0 gewählt, der ebenfalls als Ausgang low-geschaltet wird. Damit bleiben noch vier Ausgänge am Port B als potenzielle Schaltausgänge für empfangene Daten.
Beim Senden werden zwei Kommandos unterstützt. Drückt man den Taster S1 auf dem Extension-Shield, dann wird der Fernbedienungs-Tastencode „2“ gesendet. Am Empfänger wird damit der Ausgang eingeschaltet und die LED2 müsste angehen. Mit der Taste S2 sendet man eine „0“ und schaltet den Ausgang am Empfänger wieder aus.

Arduino und IR

Für die Programmierung mit Arduino-C wird die Library IRremote eingebunden (**Listing 6)**. Sie verwendet den Timer2 und seinen Ausgang OC2B (PD3, Arduino-Pin 3) für die Impulserzeugung zur Ausgabe an die IR-LED. Das wäre auch die E-Leitung des LC-Displays auf dem Extension-Shield. Deshalb kann das Programm leider nicht gleichzeitig mit dem LCD arbeiten. Das ist aber halb so schlimm, denn man hat ja noch die seriellen Ausgaben, um die empfangenen Daten zu betrachten.
Der Eingang ist beliebig und auch hier wieder auf A2 gelegt. Eine Programmierung eines Eingangs-Pullup wird durch die Library überschrieben, bleibt also unwirksam. Sie ist aber auch nicht nötig, weil der Empfang auch bei offenem Eingang im Low-Zustand das Programm nicht blockiert. Das Ziel bleibt also erreichbar, dass ein Programm ohne Änderungen oder Umschaltung wahlweise nur als Empfänger, nur als Sender oder für beide Aufgaben verwendet werden kann.

Der entscheidende Vorteil der IRremote-Library ist, dass nicht nur RC5, sondern auch eine Vielzahl anderer Normen beherrscht werden. Meist hat man ja eine ganze Sammlung unterschiedlicher Fernbedienungen im Haus. Da lohnt es sich, alle einmal auf den IR-Empfänger zu richten. Man erhält dann eine Nachricht zur verwendeten Norm und die empfangenen Daten. RC5 wird als Typ 3 geführt. Drückt man mehrfach auf die Taste 2 einer RC5-kompatiblen Fernbedienung, erscheinen folgende Meldungen.

```
3
382
3
B82
```

Die Daten werden als 12-bit-Zahl ausgegeben. Die unteren 5 Bits enthalten den Tastencode. Dazu kommt die 5-bit-Geräteadresse, in diesem Fall die Adresse 14 für einen DVBT-Receiver. Und das höchstwertige Bit ist das Toggle-Bit, das bei jeder Tastenbetätigung wechselt und nicht ausgewertet zu werden braucht. Es lohnt sich, diese Daten zu notieren, denn man kann sie dann genauso zurücksenden. Dabei darf man das Togg-

le-Bit vernachlässigen. Sendet man 382_{hex} über die IR-Diode, entspricht das der Taste „2“ im RC5-Code.
Weil der Arduino in diesem Fall ohne das Extension-Shield auskommen muss, wurde der Schaltausgang auf den Anschluss 13 gelegt. Damit wird die LED auf dem Arduino angesteuert. Am Portpin 13 kann nun das Relais angeschlossen werden, und schon kann ein Verbraucher über eine RC5-Fernbedienung oder einen zweiten Arduino mit IR-Diode geschaltet werden.

In Gegenrichtung werden auch hier wieder die Tastenkommandos 2 und 0 gesendet. Die zugehörigen Tasten liegen wie in der Bascom-Version an A0 und A1, wobei die internen Pullups eingeschaltet werden. Hier kann man vieles anschließen, was das Sensorkit bereithält. Es muss ja nicht immer ein Tastschalter sein. Warum nicht ein Magnetsensor, ein Lagesensor oder ein Lichtsensor. Da könnte man unendlich viele Ideen realisieren. Spätestens bei Sonnenaufgang wird automatisch der Fernseher abgeschaltet, oder der Lagesensor an der Türklinke erkennt, wenn jemand ins Zimmer kommt und schaltet dann zur Begrüßung das Licht und das Radio ein. ⏮

(160210)

Weblinks

[1] www.elektor.de/arduino-sensor-kit

[2] www.elektormagazine.de/160152

[3] www.elektormagazine.de/160173

[4] www.elektormagazine.de/160210

[5] www.elektormagazine.de/140009

Listing 5. RC5-Sender und -Empfänger mit Bascom.

```
'RC5LCD  In AD2, PORTC.2, Out OC1A, PORTB1
...
Do
  If S2 = 0 Then
    Togbit = 0 : Address = 0 : Command = 2
    Do
      Rc5send Togbit , Address , Command
      Waitms 100
    Loop Until S2 = 1
  End If
  If S1 = 0 Then
     Togbit = 0 : Address = 0 : Command = 0
     Do
      Rc5send Togbit , Address , Command
      Waitms 100
    Loop Until S1 = 1
  End If
  Getrc5(address , Command)
  If Address < 255 Then
    Locate 1 , 1
    Lcd Address ; "   "
    Locate 2 , 1
    Togbit = Command / 128
    Lcd Togbit ; " "
    Locate 2 , 5
    Command = Command And &B01111111
    Lcd Command ; "   "
    Print Command
    If Command = 2 Then Portb.2 = 1
    If Command = 0 Then Portb.2 = 0
  End If
Loop
```

Listing 6. IR-Steuerung in Arduino-C.

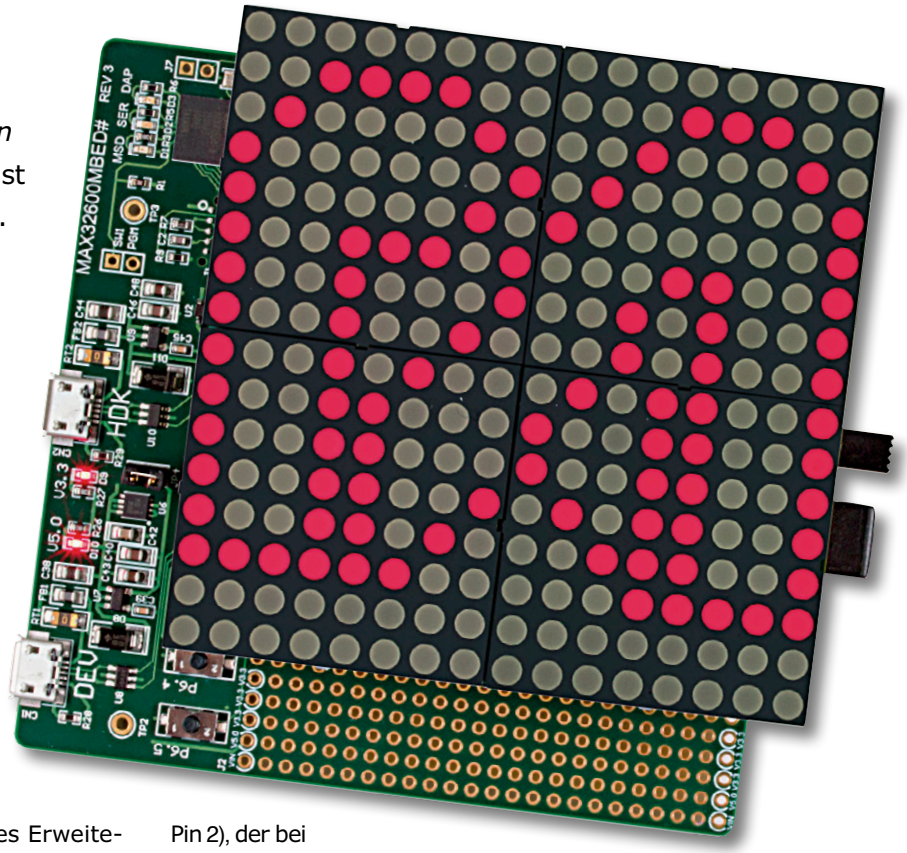
```
#include <IRremote.h>
int RECV_PIN = A2;
IRrecv irrecv(RECV_PIN);
IRsend irsend;
decode_results results;
int d;
int S1 = A0;
int S2 = A1;
int LED = 13;
int kathode =2;
void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn();
  pinMode(S1, INPUT_PULLUP);
  pinMode(S2, INPUT_PULLUP);
  pinMode(RECV_PIN, INPUT_PULLUP);
  pinMode(LED, OUTPUT);
  pinMode(kathode, OUTPUT);
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.decode_type);
    Serial.println(results.value, HEX);
    d = results.value & 15;
    Serial.println(d);
    if (d==2) digitalWrite(LED,1);
    if (d==0) digitalWrite(LED,0);
  }
  if (digitalRead(S1) == 0) irsend.sendRC5(0x382,
32);
  if (digitalRead(S2) == 0) irsend.sendRC5(0x380,
32);
  irrecv.enableIRIn();
  delay(100);
}
```

MAXREFDES99# Display Driver Shield

256 LEDs zu Ihrer Verfügung

Von **Clemens Valens** (Elektor-Labor)

Der *American Standard Code for Information Interchange* wurde 1963 veröffentlicht und ist seitdem unter dem Akronym ASCII bekannt. Er basiert auf dem englischen Alphabet, das auch bei 5 x 7 Pixel oder noch weniger gut aussieht. Heute muss ein Display alles - von arabischen Zeichen bis zu Zulu - darstellen können und benötigt deshalb mehr Pixel pro Zeichen. Bei MAXREFDES99# handelt es sich um eine einfache anzuwendende 256-Pixel-Lösung, die das kann.



Beim Shield...

...RD99 handelt es sich um ein Arduino-kompatibles Erweiterungs-Board mit einer Matrix aus 16 x 16 LEDs (faktisch vier Arrays mit 8 x 8 LEDs) samt Treiberelektronik. Das quadratische Board mit einer Kantenlänge von 7,7 cm (3") ist für Signalanwendungen gedacht, und mit den 256 LEDs kann man wohl jedes Schriftzeichen darstellen. Es kann auch Nachrichten mit einem 5x7-Font oder anderen Fonts scrollen. Was es anzeigt, hängt von Ihnen ab. Das Shield wird durch Open-Source-Software-Libraries (C++) für Arduino- und ARM-mbed-Entwicklungsplattformen unterstützt. Die Libraries kümmern sich nicht nur um die Low-Level-Kommunikation mit dem Display, sie bieten auch High-Level-Funktionen für die Anzeige aller druckbarer ASCII-Zeichen vom Leerzeichen (32) bis hin zur Tilde „~" (126) und für scrollenden Text.

Das Shield benötigt 5 V; abhängig von SW1 entweder vom Host-System oder von der Board-Versorgung. Die Eingangsspannung für den Regler auf dem Board ist durch die Maximalspannung an Pin V_{IN} (Steckverbinder H3, Pin 8) des Host-Systems bzw. von der zulässigen Spannung von C1 (25 V) begrenzt – der niedrigste Wert gilt. Für Arduino Uno R3 sind dies 20 V, doch empfohlen ist ein maximaler Wert von 12 V. Das Kit wird mit einem Steckernetzteil (für amerikanische Steckdosen) mit 9 V und 1,3 A geliefert, was schade ist, denn das Netzteil akzeptiert Netzspannungen von 100 bis 240 V. Das Shield überwacht den Pin IOREF (H3, Pin 2), der bei Arduino-Systemen die I/O-Spannung des Host-Systems signalisiert. Ein Pegelwandler passt dies automatisch an und erlaubt so I/O-Pegel bis hinab zu 1,2 V. Die Kommunikation mit dem Host-System benötigt drei Signale: Clock (ansteigende Flanke von „D13" = H4, Pin 5), Data („D11" = H4, Pin 7) und ein Ladesignal (ansteigende Flanke von „D10" = H4, Pin 8). Dies ist mit dem 16-bit-SPI-Bus kompatibel, bei dem Slave Select (SS) als Ladesignal genutzt werden kann. Hinzu kommt noch eine Reset-Leitung (active low; H3, Pin 3), IOREF und die Stromversorgung. Insgesamt sind acht Leitungen für das Interface nötig.

MAX7219

Auf dem Shield befinden sich vier LED-Display-Treiber des Typs MAX7219 – je einer für eine Matrix aus 8 x 8 LEDs. Jeder Treiber kann bis zu acht Siebensegment-Displays (mit Dezimalpunkt) = insgesamt 64 LEDs ansteuern. Jedes IC enthält einen BCD-Code-B Decoder, einen Multiplexer, Reihen- plus Spaltentreiber und ein statisches RAM mit 8 x 8 bit für die Speicherung des Zustands jeder LED. Über eine serielle Schnittstelle werden das IC programmiert und die einzelnen LEDs individuell ein- oder ausgeschaltet, ohne gleich das ganze Display updaten zu müssen. Der integrierte BCD-Decoder spart unter Umständen Arbeit, kann aber bei Nichtgebrauch auch deaktiviert werden.

Die Helligkeit aller LEDs wird vom Strom durch einen Widerstand bestimmt – bei vier Treibern also vier Widerstände. Die Helligkeit kann zudem auch via PWM mit Hilfe eines Intensity-Registers modifiziert werden.
Die Treiber werden über ein serielles Interface mit vier Leitungen an einen Mikrocontroller angeschlossen. Das serielle Protokoll entspricht dabei weitgehend dem 16-bit-SPI. Falls der Mikrocontroller das nicht kann, weil sein SPI z.B. nur 8 bit beherrscht, können die Daten auch per „bit-banging"-Technik in die Chips geschoben werden, da kein striktes Timing vorgeschrieben ist. Man muss dabei lediglich beachten, dass die Daten abhängig vom Status des LOAD-Pins das interne Shift-Register des Max7219 passieren. Eine steigende Flanke an diesem Pin überträgt die Daten des Registers nämlich in den internen Speicher und zeigt das Resultat mit den angeschlossenen LEDs an. Man muss die Daten daher zur richtigen Zeit übertragen.

Einsatz mit Arduino

Im Folgenden wird angenommen, dass eine funktionierende Arduino-Entwicklungsumgebung sowie ein Arduino Uno R3 zur Verfügung stehen. Bevor das RD99-Shield auf das Uno-Board gesteckt wird, sollte man prüfen, ob das Shield auch funktioniert. Hierzu schließt man ein Netzteil an und SW1 kommt in Richtung Eingangsbuchse, was das Board mit externer Stromversorgung verbindet. Unabhängig vom SW1 sollte eine Indikator-LED aufleuchten. Nun entfernt man das Netzteil und steckt das Shield auf das Arduino–Board, ohne SW1 zu verändern. Anschließend steckt man das Netzteil ein und verbindet den Uno mit einem PC. Bei [2] klickt man den Tab *Design Resources* und saugt die Firmware für Arduino. Anschließend entpackt man die Archiv-Datei und kopiert das Verzeichnis `MAX7219` in das Verzeichnis `Libraries` von Arduino Sketchbook (den Ablageort findet man im Arduino-Menü *File -> Preferences*). Wenn das Verzeichnis `Libraries` noch nicht existiert, legt man es an.
Nach Start der Arduino-IDE geht man zu *File -> Examples -> MAX7219* und öffnet `MAXREFDES99_example`. Wenn dieses Beispiel nicht sichtbar ist, dürfte die Library nicht korrekt installiert sein, oder die Arduino-IDE lief schon, während man die Library kopiert hat. Bei Letzterem beendet man die IDE und startet sie neu. Jetzt klickt man auf *Upload* um das Programm in den Uno zu laden. Bei mir klappte der Upload nur dann, wenn das Board vom externen Netzteil versorgt wurde.
Nach dem Upload öffnet man den *Serial Monitor* der Arduino-IDE und setzt die Baudrate auf 115.200 baud. Jetzt sollte ein Menü erscheinen. Falls nicht, betätigt man Reset beim Uno. Nun tippt man „6" in die *Send-Box* und klickt auf *Send*. Zunächst scheint nichts zu passieren, doch nach etwa 5 s wird der Befehl doch bestätigt und das Display aktiv. Man kann jetzt eine tolle Demo bewundern.
Wenn die Demo endet, erkennt der aufmerksame Beobachter

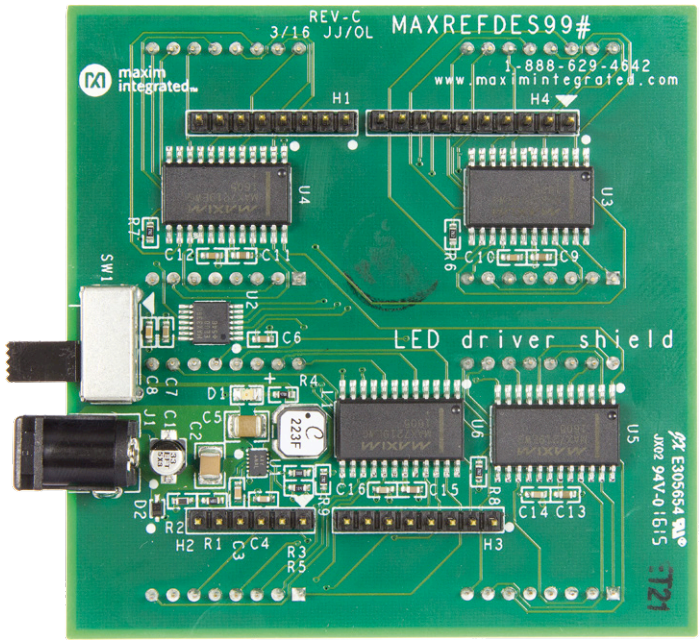


Bild 1. Rückseite des Shields.

ein Problem beim letzten Zeichen. Das „m" sieht so aus, als ob ein Pixel aus der Reihe tanzt, was aus dem „m" ein merkwürdiges Zeichen macht. Ein Blick auf das Zeichen „m" in der Datei `maxrefeds99.cpp` (schon in der Arduino-IDE geöffnet) zeigt warum: Die Sequenz in Zeile 130 „`0x58, 0x44`" sollte eigentlich „`0x78, 0x04`" sein (Wissen Sie, warum? Schreiben Sie alle Zahlen als binäre Folge, wobei 1 leuchtet und 0 dunkel ist). Korrigieren Sie den Eintrag und laden Sie die Software neu hoch. Demo #6 sollte jetzt korrekt ablaufen...

Eine kleine Animation

Das zweite Beispiel ist `MAX7219_example`. Nach Öffnen und Hochladen sieht man, was es macht. Man schaue sich den Source-Code an. Er ist sehr einfach gestrickt und demonstriert, wie man einzelne Pixel am richtigen Ort ansteuert. Inspiriert davon habe ich eine kleine Animation erstellt, die man von [3] herunterladen kann. Die Animation umfasst elf Frames, die nacheinander mit einer Pause von je 25 ms angezeigt werden und dann von vorne beginnen. Der letzte Frame wird für 1,5 s angezeigt. Ein zweiter Sketch demonstriert, wie man große eigene Zeichen darstellen kann. Unter [3] sind auch Videos zu finden.
Nachdem Sie sich die Beispiele angeschaut haben, sollten Sie in der Lage sein, Signalanwendungen nach eigenen Bedürfnissen zu erstellen. Viel Spaß dabei! ⏮

(160267)

Als autorisierter Distributor für Maxim hat Mouser mehr Produkte von Maxim im Programm als andere Distributoren. Wenn das nicht reicht: wir fügen praktisch täglich neue Produkte hinzu, sodass Sie immer die neuesten Produkte von Maxim vorfinden. Außerdem bietet Mouser Versand am gleichen Tag und flankiert eine schnelle und akkurate Belieferung mit Qualitäts-Service und Support. Werfen Sie einen Blick auf die neuesten Maxim-Produkte bei Mouser.com.

Weblinks

[1] www.mouser.com/new/maxim-integrated/maxim-maxrefdes99
[2] www.maximintegrated.com/MAXREFDES99
[3] www.elektormagazine.com/labs/maxrefdes99-led-shield-experiments-160267

Swiss Pi Extensions

Software-Beispiele

Von Peter S'heeren und Ilse Joostens (B)

Unser Erweiterungsboard „Swiss Pi" ergänzt den beliebten Einplatinencomputer Raspberry Pi um eine Reihe nützlicher Funktionen. In dieser ersten Fortsetzung des Beitrags zum Swiss Pi vom September 2016 betrachten wir den Swiss Server etwas näher, und wir beschreiben detailliert die in Python oder PHP geschriebenen Programmbeispiele.

Das Swiss Pi [1] erweitert den Raspberry Pi um eine ganze Reihe von Hardware-Funktionen: GPIO-Leitungen, PWM-Kanäle, Steuerung von Servo-Motoren, RS485, A/D-Kanäle und Echtzeituhr. Ein wichtiges Bindeglied zwischen Swiss Pi und Raspberry Pi ist eine Software, die dem Anwender den unkomplizierten und auch parallelen Zugriff auf diese Funktionen erlaubt. Das Programm *Swiss Server* arbeitet ähnlich wie ein HTTP-Server, es erlaubt Clients wie dem Anwender und fremden Programmen, die Funktionen des Boards über Instruktionen zu steuern. Mehrere Clients können zeitgleich auf das Swiss Pi zugreifen. Im vorliegenden Beitrag erklären wir zuerst, wie Sie den Swiss Server starten. Dann zeigen wir, wie Sie als Client mit dem Server interaktiv umgehen können, um das Swiss Pi zu nutzen. Ferner erklären wir, wie die Programme, geschrieben in Python oder PHP, mit dem Server kommunizieren, um auf das Swiss Pi zuzugreifen.

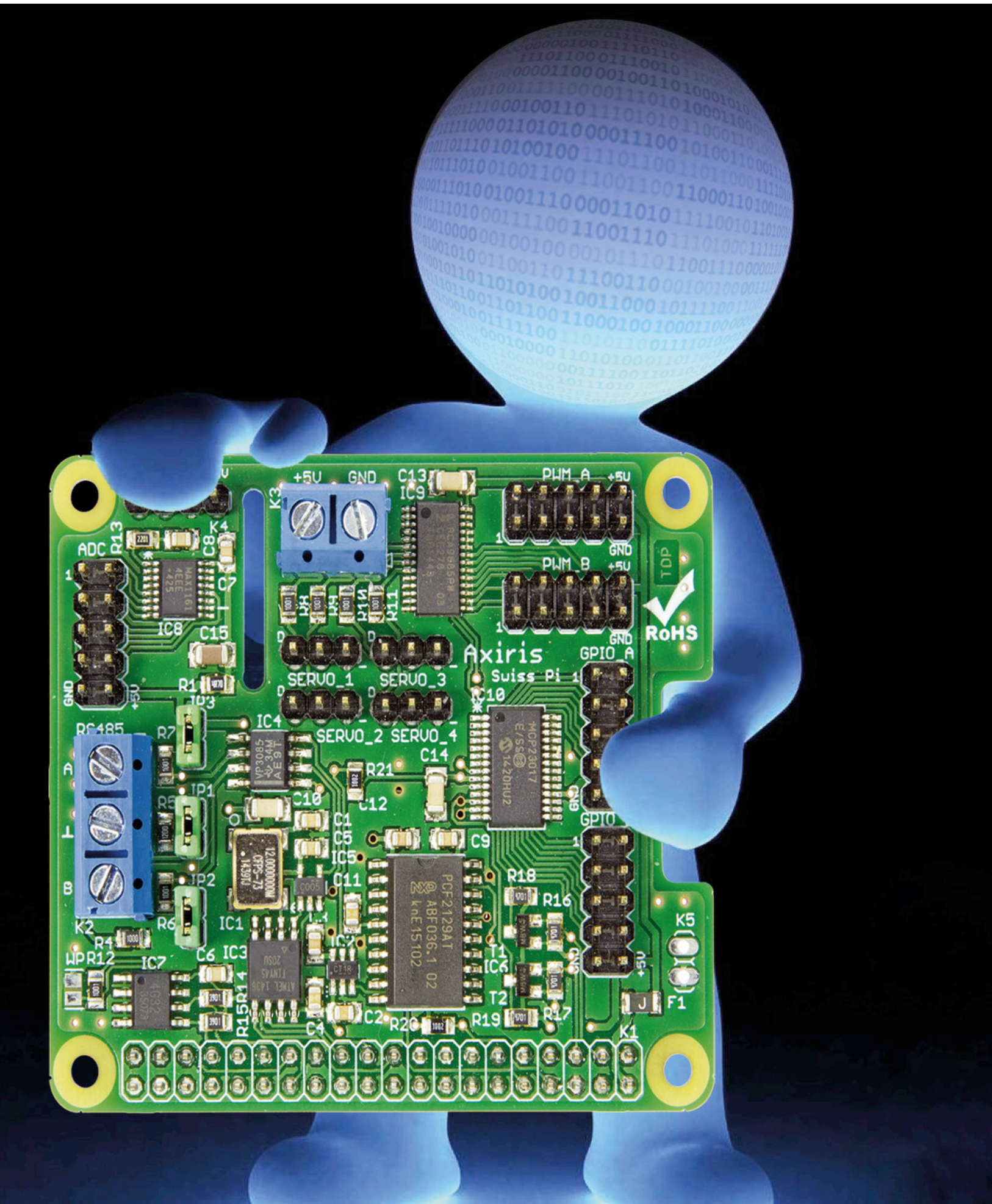
Software

Auf der Projektseite des Swiss Pi [2] haben wir die Dokumentation sowie einen Link zur Software-Seite [3] bereitgestellt. Die Software umfasst den Quellcode des Swiss Server und darüber hinaus Programme in Python oder in PHP einschließlich der dazugehörigen Dokumentation.

Swiss Server

Swiss Server ist ein Programm, das die Funktionalität des Swiss Pi über Netzwerk-Ports und per Standard-I/O in Form eines Kommunikationsprotokolls anbietet. Das Kommunikationsprotokoll besteht aus Kommandos und korrespondierenden Antworten. Der Client sendet ein Kommando, und der Server gibt eine Antwort zurück, sobald das Kommando ausgeführt ist. Clients können mehrere Kommandos unmittelbar aufeinander folgend senden, die Effizienz wird dadurch spürbar gesteigert. Das Kommunikationsprotokoll basiert auf ASCII-Zeichen. Die Syntax ist so gestaltet, dass es sowohl manuell als auch durch Software unkompliziert anwendbar ist. In der Dokumentation der Software [3] wird das Kommunikationsprotokoll ausführlich beschrieben.

Der primäre Host des Swiss Pi ist ein Raspberry Pi. Wenn das Swiss Pi mit einem AxiCat [4] (**Bild 1**) kombiniert wird, kann ein Linux- oder Windows-PC als Host eingesetzt werden. Dieser Beitrag konzentriert sich hauptsächlich auf den Raspberry Pi als Host.


+5V
GND
C13
IC9
PWM_A
PWM_B
ADC
R13
K4
C8
C7
IC8
C15
R1
IC3
IC4
RS485
R7
IP1
R5
IP2
R6
K2
R4
WP
R12
IC7
C6
IC3
R15
R14
C4
C2
R20
R19
R17
T2
IC6
T1
R18
R16
C9
C11
C5
C1
C10
C12
R21
C14
C10
SERVO_1
SERVO_3
SERVO_2
SERVO_4
Axiris
Swiss Pi
GPIO_A
GPIO
TOP
RoHS
K5
F1
K1
PCF2129AT
ABF036.1 02
knE15102
MCP23017
E/SS
1420HU2
VP3085
34M
AE9T
CFPS-73
1439TJ
ATMEL 1436
TINYAS
20SU
4G32
MAX1161
4EEE
425

Swiss Pi verbindet einen RS485-Bus mit dem UART des Host.

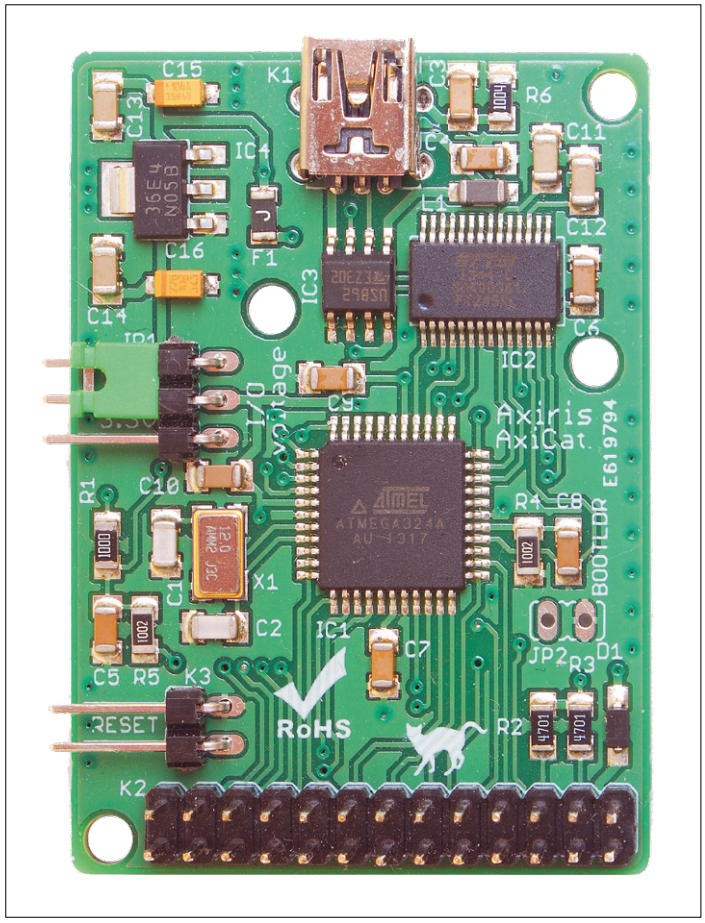


Bild 1. I²C- oder SPI-Schnittstelle an einem Linux- oder Windows-PC gefällig? Nehmen Sie einen AxiCat!

Starten

Beispiele für das Starten des Servers mit Raspberry Pi als Host:

```
./swissserver -v -i2cdev /dev/i2c-1 -p 5003

./swissserver -v -bscdetect -spidev /dev/spidev0.0
   -serial /dev/ttyS0 -rs485 spi -crlf -p 5003 -stdio
   -sp 5004
```

Beispiel für das Starten des Servers mit AxiCat und Windows-Host:

```
swissserver.exe -v -console -axicat \\.\COM12 -rs485
   spi -crlf -p 5003 -stdio -sp 5004
```

Der Parameter –v steht für *verbose*. Damit gibt der Server die Namen der benutzten Hardware-Schnittstellen und andere nützliche Informationen aus. Auf dem Bildschirm erscheinen auch die vom Server verarbeiteten Kommandos und Antworten. Diese Informationen sind nützlich, wenn der Anwender mit Python oder PHP arbeitet, denn sie geben Aufschluss über die korrekte Ausführung.

Der Parameter –console ist spezifisch für Windows, er öffnet eine Konsole, in der die Informationen des Servers ausgegeben werden.
Achtung! Die verwendeten Device-Pfade für Linux wie /dev/ttyS0 sind normalerweise in den aktuellen Versionen von Raspbian und dem Raspberry Pi Modell 3 korrekt. Die Existenz und die Bezeichnungen dieser Pfade sind jedoch im Lauf der Jahre verschiedentlich geändert worden. Falls auf dem betreffenden Raspberry Pi bestimmte Pfade nicht existieren, müssen die Einstellungen für I²C, SPI und UART in raspi-config überprüft werden. Im Internet sind mehr Informationen zu diesem Thema zu finden.

RS485

Das Swiss Pi kann eine unmittelbare Brücke zwischen einem RS485-Bus und dem UART des Host herstellen. Der RS485-Controller ist mit dem SPI-Bus des Host verbunden. Um dem Server die volle Kontrolle über RS485 zu übergeben, müssen folgende Parameter verwendet werden:

- **-serial /dev/ttyS0**: Der Server steuert den UART des Host.
- **-spidev /dev/spidev0.0**: Der Server steuert den SPI-Bus des Host.
- **-rs485 spi**: Der Server steuert den RS485-Controller des Swiss Pi.

Um Daten über den RS485-Bus zu transportieren, stehen drei Methoden zur Wahl:
- Mit den Instruktionen **serr** und **serw**. Der Server wird mit **-serial** gestartet.
- Über einen Netzwerk-Port (beispielsweise 5004). Start des Servers mit **-sp** und **-serial**.
- Über den UART des Host (beispielsweise /dev/ttyS0).

Bei der dritten Methode steuert der Server den RS485-Bus nicht direkt über den RS485-Controller an. Trotzdem muss der Controller mit dem Kommando **rste** korrekt konfiguriert werden, so dass UART und Controller die gleichen seriellen Einstellungen respektieren.

Interaktiv

Zum Kennenlernen des Servers empfehlen wir, zuerst einige Instruktionen interaktiv zu testen. Wenn der Server mit **-stdio** gestartet wird, können die Instruktionen unmittelbar in der Shell eingegeben werden. Wird der Server mit **-p** gestartet, kann eine Verbindung mit einem Terminalprogramm wie PuTTY, Netcat oder Hyperterminal aufgebaut werden. Für die Eingabe von Instruktionen steht dann das Terminalfenster zur Verfügung. Beispiel: Verbinden mit Netcat auf dem Raspberry Pi: $ nc localhost 5003.

In den nachfolgenden Bespielen sind die Instruktionen in Schwarz, die Antworten des Servers in Blau dargestellt. Folgende Instruktion gibt die Version des Servers aus:

```
ver
ver "1.0.1"
```

Optional kann jeder Instruktion eine Identifikationsnummer hinzugefügt werden, sie wird zusammen mit der Antwort zurückgegeben. Beispiel: Warte 100 ms:

```
id 10 wait 100
id 10 wait ok
```

Optional kann jeder Instruktion ein Präfix vorangestellt werden, um die Antwort zu unterdrücken:

```
norsp wait 100
```

GPIO

Eine GPIO-Leitung kann als Eingang konfiguriert werden. Beispiel: Setze GPIO-Leitung 10 als Eingang, aktiviere die Pullup-Widerstände und lies das Signal am Eingang:

```
iod 10 1
iod ok
iopu 10 1
iopu ok
ior 10
ior 10 1 0 1 1
```

Der unterstrichene Wert in der letzten Antwort gibt an, dass das Signal High ist.

Eine GPIO-Leitung kann auch als Ausgang konfiguriert werden. Beispiel: Setze GPIO-Leitung 11 als Ausgang, setze den Ausgang auf High:

```
iod 11 0
iod ok
iow 11 1
iow ok
```

ADC

Mit der Instruktion **adcr** werden die Signale der acht A/D-Kanäle konvertiert und die Ergebnisse in der Antwort zurückgegeben:

```
adcr
adcr 3460 1843 0756 0233 0060 0000 0000 0000
```

Der Server speichert die Ergebnisse der letzten Konvertierung. Die Ergebnisse können jederzeit mit der Instruktion **adcrc** abgerufen werden. Im nächsten Beispiel werden die gespeicherten Ergebnisse für die A/D-Kanäle 2 bis 4 gelesen:

```
adcrc 2 3
adcrc 2 3 0756 0233 0060
```

PWM

Der PWM-Controller hat ein Prescaler-Register für die PWM-Frequenz sowie 16 Kanäle mit jeweils vier Parametern, um die relative Startzeit und die Periode des Rechtecksignals zu steuern. Die Bedeutungen der Parameter werden in der Dokumentation erklärt. Mit den Instruktionen **ppr** und **ppw** kann das Prescaler-Register gelesen oder beschrieben werden, mit den Instruktionen **pcr** und **pcw** ist das Lesen oder Schreiben eines oder mehrerer PWM-Kanäle möglich:

```
ppr
ppr 030
ppw 121
ppw ok
pcr 5
pcr 05 01 0 0000 1 0000
pcw 9 1 0 1250 0 1280
pcw ok
pcr 8 2
pcr 08 02 0 0000 1 0000 0 1250 0 1280
```

Diese Instruktionen bewirken folgendes: Lies das Prescaler-Register, setze das Register auf 121 (50 Hz), lies PWM-Kanal 5, schreibe PWM-Kanal 9, lies die PWM-Kanäle 8 und 9. In der Dokumentation werden mehrere Varianten von **pcr** und **pcw** beschrieben.

Servo

Der Server hat eine eingebaute Servosteuerung, die jeden der 16 PWM-Kanäle individuell steuern kann. Die PWM-Kanäle 0 bis 3 sind auf separate Anschlüsse auf dem Swiss Pi herausgeführt.

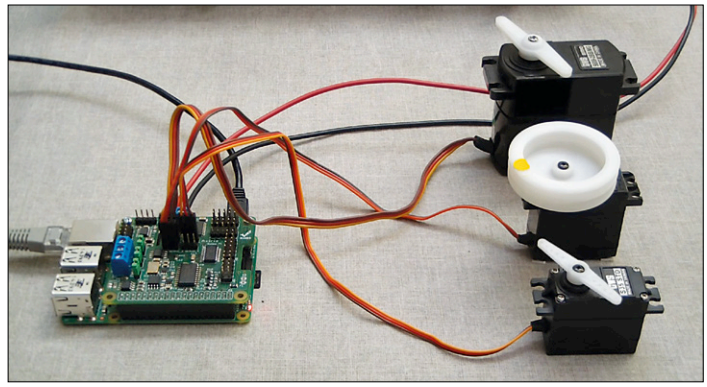


Bild 2. Das Steuern von Servos wird zum Kinderspiel.

Die Servosteuerung kann ein- und ausgeschaltet werden (**svme**, **svmd**), PWM-Kanäle sind als Servokanäle konfigurierbar (**svcw**), und das Abfragen des Status ist möglich (**svcr**).

Um Servomotoren zu steuern, muss die PWM-Frequenz auf 50 Hz gesetzt werden:

```
ppw 121
ppw ok
```

Für das Konfigurieren eines PWM-Kanals benötigt **svcw** folgende Informationen:

- Kanalnummer (0...15)
- Niedrigste zulässige 12-bit-Impulsbreite (0...4095)
- Höchste zulässige 12-bit-Impulsbreite (0...4095)
- Art der Motorpositionssteuerung: In Schritten (0...max), oder 12-bit-Impulsbreite (0...4095)

Für Servomotoren müssen Sie die PWM-Frequenz auf 50 Hz setzen.

```
svcw 0 118 515 1000
svcw ok
```

Diese Instruktion legt fest, dass die Servosteuerung PWM-Kanal 0 mit Impulsbreiten von 118 bis 515 steuern kann und dass ein Schrittwert von 0 bis 1000 an **svmv** übergeben werden soll, um Impulsbreiten von 118 bis 515 einzustellen.

```
svcw 1 122 520 0
svcw ok
```

Diese Instruktion legt fest, dass die Servosteuerung PWM-Kanal 1 mit PWM-Perioden von 122 bis 520 steuern kann und dass eine Impulsbreite an **svmv** übergeben werden soll.

Bevor ein Motor gesteuert werden kann, muss die Servosteuerung aktiviert werden:

```
svme
svme ok
```

Jetzt lässt sich die Motorposition verändern.

```
svmv 0 500 0 0
svmv 00 ok
```

Diese Instruktion setzt die Motorposition auf Schritt 500, also in die Mitte zwischen Schritt 0 und 1000. Der Server setzt diesen Schrittwert in die Impulsbreite 316 um, in die Mitte zwischen 118 und 515.

```
svmv 1 400 0 0
svmv 01 ok
```

Diese Instruktion setzt die Motorposition auf die Impulsbreite 400.

Der Motor bewegt sich unmittelbar zur angegebenen Position. Es ist möglich, eine Periode für die Bewegung einzustellen:

```
svmv 0 500 2500 0
svcr 0
svcr 00 0118 0515 1000 0174 0141 1
svcr 0
svcr 00 0118 0515 1000 0257 0350 1
svmv 00 ok
svcr 0
svcr 00 0118 0515 1000 0316 0498 0
```

Die erste Instruktion führt die Motorbewegung über eine Periode von 2500 aus. Die **svcr**-Instruktionen geben die aktuelle Motorposition zurück. Der Server sendet die **svmv**-Antwort erst, wenn die Motorbewegung abgeschlossen ist. Die Instruktion **svmv** muss im Hintergrund ausgeführt werden, sie ist eine asynchrone Instruktion.
Es ist möglich, mehrere **svmv**-Instruktionen für den gleichen Kanal nacheinander zu senden:

```
svmv 0 250 2000 0
svmv 0 500 2000 0
svmv 0 400 1000 0
```

Die **svmv**-Instruktionen werden nacheinander ausgeführt. Die Gesamtzeit beträgt mindestens 5000 ms, da zwischen den Instruktionen Verzögerungen auftreten können. Wenn die Motorbewegung während exakt 5000 ms fließend ablaufen soll, ist der vierte Parameter von **svmv** relevant. Dieser Parameter gibt an, wie viele Millisekunden nach dem Start der letzten **svmv**-Instruktion diese **svmv**-Instruktion starten darf. Der Server kompensiert eventuelle Verzögerungen, so dass die gewählten Zeiten der Ausführung immer eingehalten werden.

```
svmv 0 250 2000 0
svmv 0 500 2000 2000
svmv 0 400 1000 2000
```

Wenn eine vorangegangene **svmv**-Instruktion für den Kanal nicht aktiv ist, nimmt der Server **svme** als Startzeit. In diesem Fall muss dafür gesorgt werden, dass die aktuelle Impulsbreite im gültigen Bereich liegt, anderenfalls kann der Server die Bewegung von der aktuellen zur neuen Position nicht ausführen. Diese Vorgehensweise ermöglicht das Synchronisieren mehrerer Motoren:

```
svme
svme ok
svmv 0 250 2000 500
svmv 1 400 1550 500
```

Der Server startet beide Motorbewegungen 500 ms nach **svme**.

Echtzeituhr

Der Server führt Instruktionen aus, um die Echtzeituhr zu lesen (**rtcr**) und einzustellen (**rtcw**).

```
rtcr
rtcr 1 1
rtcw 2016 10 21 15 42 55
rtcw ok
rtcr
rtcr 0 1 2016 10 21 15 42 58
```

Die erste Antwort bedeutet, dass die Echtzeituhr keine gültigen Werte für Datum und Zeit zurückgeben kann und dass die Batterie nicht angeschlossen ist. Die zweite Instruktion setzt Datum und Zeit, die dritte Instruktion gibt Datum und Zeit zurück.

Mit sercfg können Sie den UART des Host und den RS485-Controller gleichzeitig konfigurieren.

RS485

Mit **sercfg** lassen sich der UART des Host und der RS485-Controller gleichzeitig konfigurieren, vorausgesetzt dass der Server mit **-serial**, **-spidev** und **-rs485 spi** gestartet wurde.
Exemplarisch haben wir einen DIN-Schienen-Energiemesser SDM120C am RS485-Bus angeschlossen. Die folgende Instruktion konfiguriert die serielle Schnittstelle mit 2400 Baud, 8 Datenbit, 1 Stoppbit, keine Parität und weist an, den RS485-Controller gleichzeitig zu konfigurieren:

```
sercfg 2400 8 1 none 1
sercfg ok
```

Der SDM120C arbeitet mit ModBus RTU. Die folgende Instruktion fragt die Spannung ab:

```
serw 01h 04h 00h 00h 00h 02h 71h 0CBh
serw ok
```

Jetzt kann die Antwort gelesen werden:

```
serr
serr 001 004 004 067 107 153 154 117 231
```

In der Dokumentation des SDM120C ist beschrieben, dass die unterstrichenen Bytes die Spannung als 32-bit-Fließkommazahl wiedergeben.

Python

Die Programmbeispiele wurden für Python 3 geschrieben. Wenn auf dem Raspberry Pi das Betriebssystem Raspbian läuft, ist Python bereits installiert. An dieser Stelle können wir nur die wichtigsten Elemente betrachten.
Damit ein Programm ausgeführt wird, muss Python mit dem Dateinamen des Programms als Parameter nach dem Kommandoprompt gestartet werden:

Linux: `python3 swisspi_gpio_read_pin.py`

Windows: `python swisspi_gpio_read_pin.py`

Die Programme benutzen den Swiss Server, um mit dem Swiss Pi zu kommunizieren. Folglich können die Programme sowohl lokal auf dem Raspberry Pi als auch auf einem anderen System ausgeführt werden.
Die Programme machen Gebrauch von `class TextLineNetClient` in *textlinenetclient.py*.
Damit wird die Verbindung zum Swiss Server hergestellt, um Instruktionen zu senden und Antworten zu verarbeiten. Der Name der Class deutet bereits darauf hin, dass die Class ein Netzwerk-Client ist, der Textzeilen mit dem Server austauscht. Die Class generiert eine Fehlermeldung des Typs `class TextLineNetClientError`, falls ein Fehler auftritt, beispielsweise eine Unterbrechung der Serververbindung.
Der folgende Python-Code baut eine Verbindung zum Swiss Server auf:

```
host = 'localhost'
port = 5003
client = textlinenetclient.
   TextLineNetClient(host,port)
```

Diese Verbindung ist lokal, der Python-Code muss auf dem Raspberry Pi laufen. Für den Zugriff von einem anderen System auf das Swiss Pi muss die IP-Adresse des Raspberry Pi in der Form `host = '192.168.1.115'` eingegeben werden. Die IP-Adresse des Raspberry Pi ist mit dem Kommando `ifconfig` abfragbar.
Die Variable `client` bezeichnet die Verbindung. Zur Class gehören Funktionen, die im Textzeilenformat Instruktionen senden und Antworten empfangen.
Geschrieben werden Instruktionen in zwei Schritten. Die Class verwendet einen Schreibpuffer, die Funktion `sendCmd` fügt eine Instruktion zum Schreibpuffer hinzu. Die Instruktionen im Schreibpuffer werden mit der Funktion `commit` zum Server gesendet. Alternativ gibt es für `sendCmd` den zusätzlichen Parameter `True`. Der Vorteil dieser Vorgehensweise liegt darin, dass mehrere Instruktionen als Gruppe zum Server gesendet werden können. Verglichen mit der einzelnen Verarbeitung ist dies deutlich effizienter.
Die Funktion `rcvRsp` liest Daten vom Server und gibt beim Aufrufen eine Antwort zurück. Diese Funktion blockiert, solange die Antwort aussteht.
Die Funktion `tokenizeLine` wandelt Textzeilen in Symbolreihen um (*token*). Die Programme benutzen diese Funktion, um Antworten des Servers auf einfache Weise zu analysieren. Wenn der Server mit dem Parameter -v gestartet wurde, können die Antworten des Servers mitverfolgt werden.
Zahlenwerte, die in Antworten vorkommen, werden in einem vorgebbaren Muster formatiert. Standardmäßig sind verschiedene Formatierungen für gute Lesbarkeit bei interaktivem Betrieb eingestellt. In Python ist es jedoch zweckmäßiger, Zahlen als dezimale Werte zu akquirieren. Dazu dient folgendes Kommando:

```
client.sendCmd('norsp vfmts "*" dec 0 0 0 0 0 0')
```

Anzumerken ist hier, dass der Server wegen `norsp` nicht antwortet. Dies ist auch nicht notwendig, um eine Instruktion unmittelbar zum Server zu senden. Wichtig ist, dass der Server diese Instruktion als erste empfängt.

GPIO

Dieser Python-Code setzt eine GPIO-Leitung als Eingang, aktiviert den Pullup-Widerstand und liest das Eingangssignal. Die Variable `pin` selektiert die GPIO-Leitung (0...15).

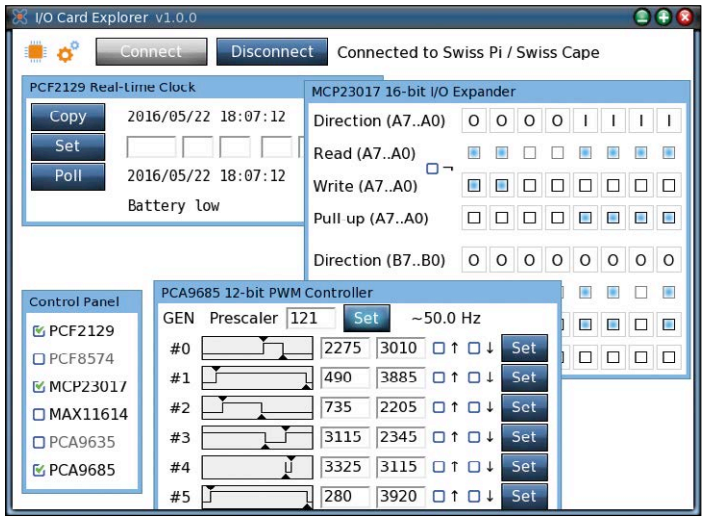


Bild 3. Mit dem I/O Card Explorer greifen Sie auf das Swiss Pi zu.

```
pin = 10
client.sendCmd('norsp iod %d 1' % pin)
client.sendCmd('norsp iopu %d 1' % pin)
client.sendCmd('ior %d' % pin,True)
tokens = client.tokenizeLine(client.rcvRsp())
print("GPIO pin %d: input state %d" %
   (pin,int(tokens[2])))
```

Mit dem I/O Card Explorer betrachten Sie die Wirkung von Instruktionen auf dem Swiss Pi.

Der Code sendet drei Instruktionen zum Server. Die erste Instruktion setzt die GPIO-Leitung 10 als Eingang, die zweite aktiviert den Pullup-Widerstand und die dritte liest das Signal. Der dritte Aufruf von `sendCmd` sendet wegen des Parameters `True` gleichzeitig den Inhalt des Schreibpuffers zum Server. Der Server antwortet auf die dritte Instruktion. Die Antwort wird in Symbole umgesetzt, das dritte Symbol (tokens[2]) enthält den Signalzustand der GPIO-Leitung.
Das Ausgangssignal einer GPIO-Leitung wird wie folgt gesetzt. Variable `pin` selektiert die GPIO-Leitung (0...15), Variable `output` enthält den Signalzustand (0 oder 1). Nachfolgend wird die als Ausgang konfigurierte GPIO-Leitung 2 auf High gesetzt:

```
pin = 2
output = 1
client.sendCmd('norsp iow %d %d' % (pin,output),True)
```

ADC

Mit der Instruktion **adcr** werden die Signale auf den acht A/D-Kanälen konvertiert und die Ergebnisse in der Antwort zurückgegeben. Die Kanäle sind von 0 bis 7 nummeriert. Im folgenden Beispiel wird der Wert von Kanal 4 gelesen:

```
ch = 4
client.sendCmd('adcr',True)
tokens = client.tokenizeLine(client.rcvRsp())
print("A/D channel %d: %d" % (ch,int(tokens[1 + ch])))
```

PWM

Der PWM-Controller hat ein Prescaler-Register zum Einstellen der PWM-Frequenz. Das Register wird wie folgt gelesen:

```
client.sendCmd('ppr',True)
tokens = client.tokenizeLine(client.rcvRsp())
print("PWM prescale: %d" % int(tokens[1]))
```

Beschreiben lässt sich das Prescaler-Register auf diese Weise:

```
pre = 121  # 50 Hz
client.sendCmd('norsp ppw %d' % pre,True)
```

Jeder PWM-Kanal kennt vier Parameter, mit denen die relative Startzeit und die Periode des Rechtecksignals gesteuert werden können. Eine einfache Methode, die Periode festzulegen, ist folgende:

```
ch = 4
on_period = 1250
client.sendCmd('norsp pcw %d 1 0 0 0 %d' %
   (ch,on_period),True)
```

Servo

Der folgende Code konfiguriert zwei Kanäle und bewegt die Motoren zu einer bestimmten Position:

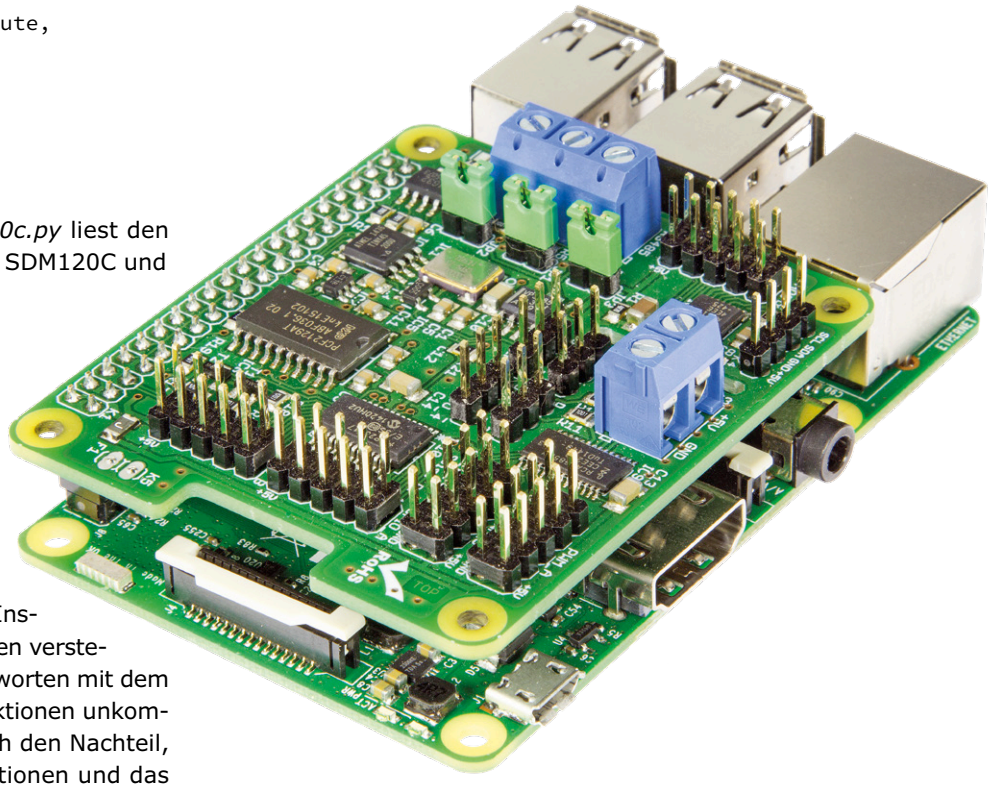
```
client.sendCmd('norsp svcw 0 118 515 1000')
client.sendCmd('norsp svcw 1 122 520 0')
client.sendCmd('norsp svme')
client.sendCmd('norsp svmv 0 250 0 0')
client.sendCmd('norsp svmv 1 400 0 0')
client.commit()
```

Echtzeituhr

Der folgende Code stellt die Echtzeituhr auf die lokale Systemzeit. Gleichzeitig demonstriert der Code die Anwendung der Funktion `commit`, sie ist äquivalent zum Parameter `True` bei `sendCmd`.

```
t = datetime.datetime.now()
client.sendCmd('norsp rtcw %d %d %d %d %d %d' %
```

```
    (t.year, t.month, t.day, t.hour, t.minute,
    t.second))
client.commit()
```



RS485

Das Programmbeispiel *swisspi_rs485_sdm120c.py* liest den Wert der Spannung aus einem Energiemesser SDM120C und gibt ihn aus.

PHP

Die Programmbeispiele benutzen den Swiss Server, um mit dem Swiss Pi zu kommunizieren. Deshalb können die Programme sowohl lokal auf dem Raspberry Pi als auch auf anderen Systemen laufen.
Die Programme machen Gebrauch von `class SwissClient` in *SwissClient.php*. Diese Class bietet eine Reihe von Funktionen an, die Instruktionen des Servers ausführen. Die Funktionen verstecken den Austausch von Instruktionen und Antworten mit dem Server. Der Vorteil besteht darin, dass die Funktionen unkompliziert einsetzbar sind. Erkauft wird dies durch den Nachteil, dass das simultane Senden mehrerer Instruktionen und das Verarbeiten asynchroner Antworten nicht möglich ist. PHP wird hauptsächlich verwendet, um Webseiten zu generieren, hier genügt der synchrone Zugang zum Swiss Pi.
Der folgende Code stellt eine Verbindung mit dem Server her. Zuerst wird ein `SwissClient`-Objekt erzeugt, danach wird die Funktion `connect` aufgerufen. Diese Funktion gibt `true` zurück, wenn der Aufbau erfolgreich war, und `false`, wenn er fehlgeschlagen ist.

```
$client = new SwissClient('localhost',5003);
$ok = $client->connect();
if (!$ok) return false;
```

Dies ist eine lokale Verbindung, der PHP-Code muss auf dem Raspberry Pi ausgeführt werden. Wenn von einem anderen System auf das Swiss Pi zugegriffen werden soll, ist die IP-Adresse des Raspberry Pi anzugeben:

```
$client = new SwissClient('192.168.1.115',5003);
```

Die Wirkungsweise der Funktionen sind in *SwissClient.php* festgelegt. Hier folgen nur einige Beispiele:

Lies den Signalzustand auf GPIO-Leitung 4 und gib das Ergebnis aus:

```
$res = $client->readIOPin(4);
var_dump($res);
```

Setze GPIO-Leitung 4 als Ausgang auf High, übertrage die Antwort des Servers:

```
$client->writeIOPin(4,true,true);
```

Schreibe nach zwei PWM-Kanälen:

```
$pwm_data = array
(
    array("always_on" => false, "on_pos" => 1200,
   "always_off" => false, "off_pos" => 2424),
    array("always_on" => true,  "on_pos" =>  444,
   "always_off" => true,  "off_pos" => 100)
);

$client->writePWM2Range(14,$pwm_data,true);
```

Schlusswort

Unser Software-Paket enthält Programmbeispiele in Python oder PHP, die Sie sich anschauen sollten. Sie können mit dem Programm *I/O Card Explorer* [5] jederzeit eine Verbindung zum Swiss Server herstellen (**Bild 3**). So können Sie beim interaktiven Umgang mit Instruktionen oder beim Ausführen von Python- oder PHP-Code die Aktivitäten auf dem Swiss Pi mitverfolgen und gegebenenfalls darauf reagieren.
In der nächsten Folge stellen wir nützliche Hardware-Erweiterungen vor, darunter sind ein 8-Kanal-Relaisboard, ein digitales 8-Kanal-Eingangsboard, eine DC-Motorsteuerung, ein Current-loop-Interface sowie ein Board, dass PWM in Spannungen 0...10 V umsetzt.

(160237)gd

Weblinks

[1] www.elektormagazine.de/150584

[2] www.axiris.eu/en/index.php/i-o-cards/swiss-pi

[3] www.axiris.eu/en/index.php/free-software/software-repository

[4] www.elektormagazine.de/150585

[5] www.axiris.eu/en/index.php/free-software/i-o-card-explorer

[6] www.elektormagazine.de/160237

Verkehrssignal-Steuerung

Für Straßenkreuzungen von Modellbahnanlagen

Von **Piet Kralt** (NL)

Zu einer Modellbahnanlage gehören nicht nur Schienen, sondern auch Straßen. Wo sich vielbefahrene Straßen kreuzen, stehen Verkehrssignale, die den Fahrzeugen wechselweise freie Fahrt gewähren. Die hier vorgestellte Verkehrssignal-Steuerung schaltet die miniaturisierten Ampeln getreu ihrem originalen Vorbild.

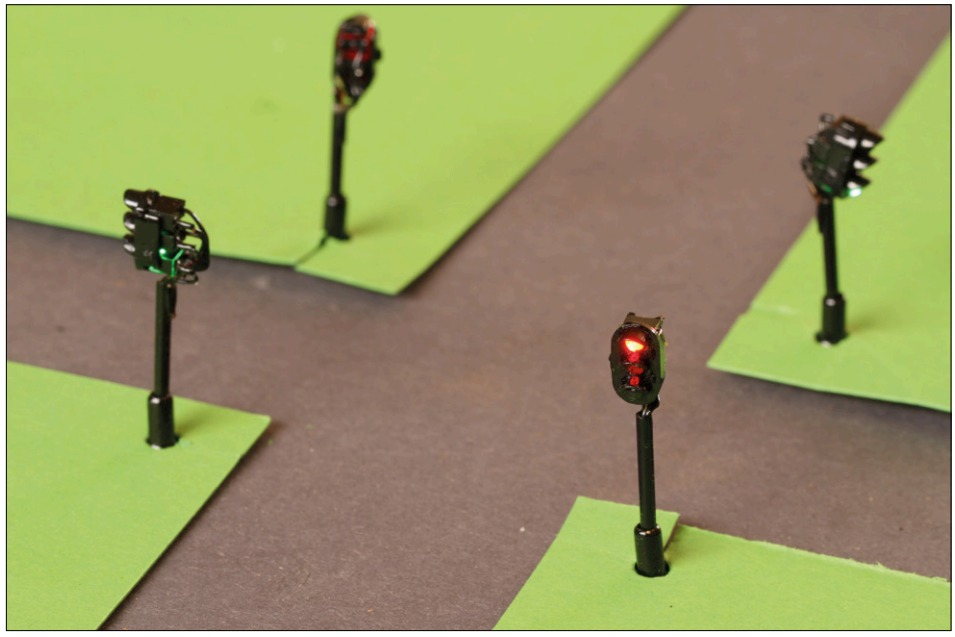


Bild 1. Bei diesem Signal sind die LED-Kathoden herausgeführt, die Anoden liegen über Widerstände an einer gemeinsamen Leitung.

Der Selbstbau von Verkehrssignalen für die Modellbahnanlage ist sicher nicht jedermanns Sache, doch im Modellbau-Fachhandel ist das Angebot riesig. Das Foto in **Bild 1** zeigt ein solches Signal für Anlagen der Spurweite N. Bei diesem Signal sind den LED-Anoden die Widerstände zur Strombegrenzung vorgeschaltet, von dort führt eine gemeinsame Leitung zum Pluspol der Stromversorgung. Mit 1 kΩ sind die Widerstände für die Betriebsspannung 12...15 V ausgelegt, der LED-Strom beträgt dann ungefähr 10...13 mA.

Schaltung

Für das Steuern von Verkehrssignalen im Modellbau existieren zahllose Schaltungen. Hier übernimmt ein Mikrocontroller aus der PIC-Familie diese Aufgabe, die Wahl fiel auf den Typ PIC16F526. Die Ausgänge dürfen bis 25 mA belastet werden, so dass jeder Ausgang zwei LEDs steuern kann. Da die Spannung des PICs nur 5 V beträgt, müssen die für 12 V gebauten Signale angepasst werden. Die 1-kΩ-Widerstände werden entfernt, an ihrer Stelle werden in die Kathodenleitungen, wie die Schaltung in **Bild 2** zeigt, Widerstände 270 Ω oder 330 Ω gelegt. Die Schaltung steuert die vier Verkehrsampeln einer regulären Straßenkreuzung. Die Farben der an einer Straße gegenüberliegenden Ampeln werden parallel gesteuert, sie liegen über die Strombegrenzungswiderstände am jeweils gleichen Mikrocontroller-Ausgang.
Weil Modellbahnanlagen in der Regel die Spannung 12 V bereitstellen, die Spannung 5 V jedoch nur selten verfügbar ist, setzt ein Spannungsregler 7805 die Spannung auf 5 V herab. Für die Programmierung des Mikrocontrollers in der Schaltung ist Steckverbinder SV1 vorgesehen. Wenn der PIC außerhalb der Schaltung mit einem Programmer programmiert wird, kann SV1 entfallen. SV2 dient zum Umschalten der Signale vom regulären Betrieb auf gelbes Blinklicht. Die Steckverbinder SV3 und SV4 sind die Anschlüsse für die LEDs der Signale, wobei jede LED einen eigenen Anschluss hat. Drucktaster S1 ist für eventuelle Erweiterungen vorgesehen.

PIC-Programm

Damit die Schaltung unkompliziert bleibt, wurde die Funktionalität in das Mikrocontroller-Programm verlagert. Programme lassen sich flexibel modifizieren, das Anpassen einer Schaltung erfordert in der Regel einen höheren Aufwand.

Für die Verkehrssignal-Steuerung wurde ein leicht überschaubares Programm in Assembler geschrieben, es kann von der Projektseite [1] frei heruntergeladen werden. In der ASM-Datei lassen sich die Ampel-Schaltzeiten anpassen, auch die Farbfolge, beispielsweise gleichzeitig Rot und Gelb, ist änderbar.

Nach der Modifizierung muss die ASM-Datei neu assembliert werden, die assemblierte Datei wird in den Programmspeicher des PIC geladen.

Prototypen-Platine

Das ZIP-Archiv 150710-11.zip, das zum Download von der Projektseite [1] gehört, enthält die CAD-Dateien eines Platinenentwurfs. Die Platinenabmessungen wurden absichtlich möglichst kompakt gehalten. Noch geringere Abmessungen wären mit SMD-Komponenten erreichbar, doch hier stehen Aufwand und Nutzen nicht in angemessenem Verhältnis.

Den aufgebauten Prototyp zeigt das Foto in **Bild 3**. Wegen des Wannensteckers sind die Komponenten auf der aktuellsten Platinenversion etwas verschoben worden. Ein einfacher Pfostenverbinder mit 2 · 5 Kontakten wäre zwar kleiner und preiswerter, er hat jedoch den Nachteil, dass die zugehörige Steckleiste versehentlich falsch aufgesteckt werden kann. Jede Signal-LED hat auf der Platine einen eigenen Anschluss, beim Prototyp wurden die Leitungen unmittelbar an die Kupferinseln auf der Platine gelötet. Ursprünglich waren hier Kabelschraubklemmen vorgesehen, sie haben sich jedoch wegen der anzuschließenden dünnen Leitungsadern nicht bewährt.

Zum Schluss

Der Prototyp der Verkehrssignal-Steuerung arbeitete wie geplant, womit das erste Ziel erreicht war. Modifizierungen und Ergänzungen liegen natürlich im Bereich des Möglichen, hier sind einige Beispiele:

- Nachschalten von Leistungstreibern, so dass größere LED-Typen oder auch Glühlampen gesteuert werden können,
- Umschalten auf Farbfolgen, wie sie in anderen Ländern gebräuchlich sind, zum Beispiel Rot → Rot/Gelb → Grün → Gelb → Rot,
- Einstellen abweichender Rot-, Gelb- und Grün-Phasen über den vorhandenen Taster S1.

Die Hard- und Software lässt sich nach Belieben anpassen, die Möglichkeiten sind vielfältig

(150710)gd

Fotos: Dirk Jan Kralt

Tabelle 1. LED-Anschlüsse		
Leitung SV3	Signal	Anschluss/ Farbe
1	1	Gruppe 1
3	1	Grün
5	1	Gelb
7	1	Rot
2	3	Gruppe 3
4	3	Grün
6	3	Gelb
8	3	Rot

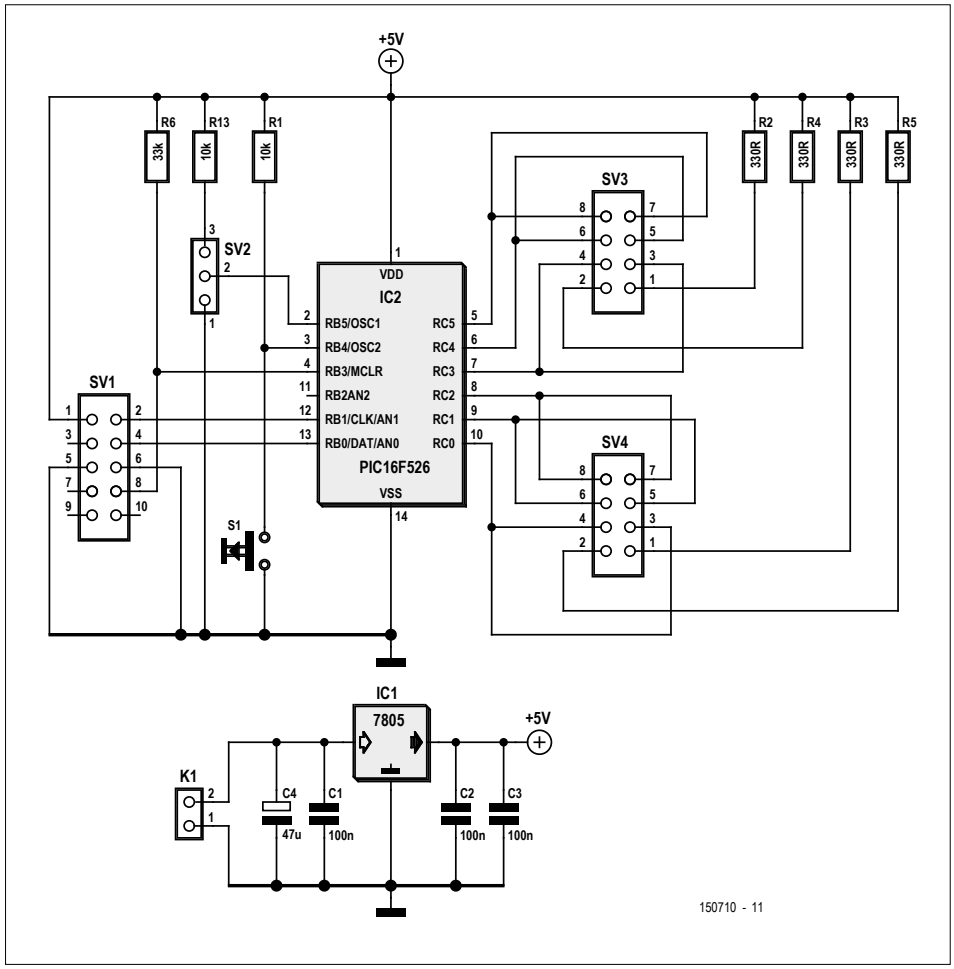


Bild 2. Die Steuerung übernimmt ein Mikrocontroller PIC16F526.

Weblinks

[1] Software- und CAD-Downloads: www.elektormagazine.de/150710

[2] Projektseite: www.elektormagazine.de/labs/stoplichtautomaat-voor-de-modelbaan

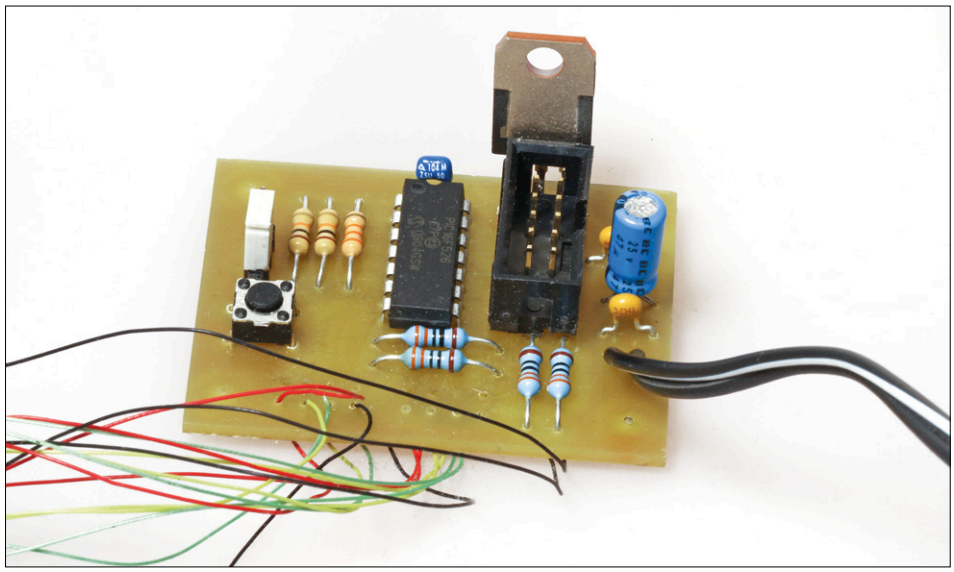


Bild 3. Die Anschlussleitungen wurden beim Prototyp an die Platine gelötet.

Antennen-Diplexer

Fürs Autoradio mit DAB+

Wenn Sie Ihr UKW-Radio im Auto durch ein moderneres Gerät mit DAB+-Empfang ersetzen wollen, werden Sie bemerken, dass so ein Exemplar zwei separate Antenneneingänge für UKW und DAB+ besitzt. Dafür sind eigentlich auch zwei einzelne Antennen notwendig. Doch mit dieser Hochfrequenzweiche ist es möglich, beide Eingänge an die Standard-Autoantenne anzuschließen.

Von **Alfred Rosenkränzer** (D)

Ich wurde auf diese bemerkenswerte Filterart aufmerksam, als ich überlegte, das schwächelnde UKW-Radio in unserem Auto durch ein moderneres Gerät mit DAB+-Empfang zu ersetzen. Unser Auto hat glücklicherweise noch einen DIN-Schacht, so dass der Austausch relativ problemlos möglich ist; entsprechende Radios sind verfügbar.

Ein DAB+-Autoradio hat allerdings zwei Antenneneingänge, einen für UKW (FM) und einen separaten für DAB+. Der für UKW wird wie gewohnt mit der Autoantenne verbunden, für DAB+ liegt einigen Geräten eine Scheibenantenne bei, falls das Auto noch nicht fürs Digitalradio vorbereitet ist (was bei unserem der Fall ist). Diese (Front-)Scheibenantenne ist optisch natürlich kein Gewinn, außerdem muss man für die Antennenleitung ein Loch im Armaturenbrett anbringen. Den unerfreulichen Diskussionen mit meiner Frau wollte ich auf jeden Fall aus dem Weg gehen.

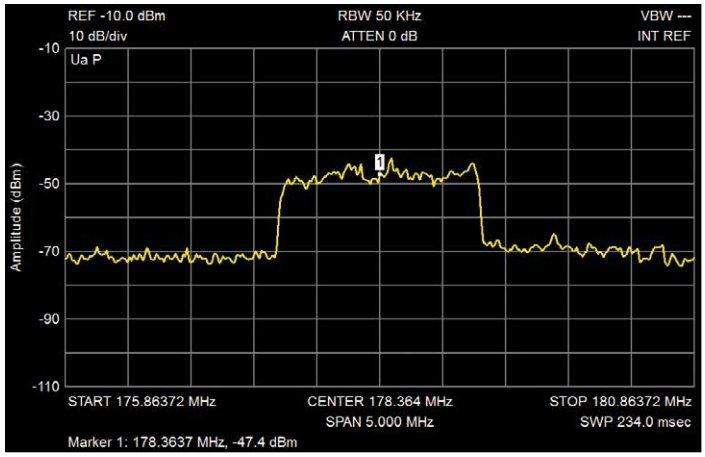


Bild 1. Das Bild eines DAB+-Spektrums bei 178 MHz, Kanal 5c.

Möchte man die serienmäßige Antenne benutzen, so muss man zunächst feststellen ob sie überhaupt ein Signal im DAB+-Bereich empfängt; einige aktive Antennen haben nämlich ein Tiefpassfilter eingebaut. Am einfachsten kann man dies mit einem Spektrumanalyser messen (**Bild 1**), ansonsten mit einem DAB+-Empfänger testen. Das UKW(FM)-Band überdeckt bekanntlich 88 bis 108 MHz, das DAB-Band III geht von 174 bis 230 MHz. Das L-Band im Bereich von 1,5 GHz spielt für die Übertragung zum Auto keine Rolle.

Das vorhandene Antennensignal müsste dann auf zwei Antenneneingänge aufgesplittet werden. Im einfachsten Fall schließt man zwei Kabel parallel an. Dies verletzt natürlich den Wellenwiderstand der Leitung und es führt zu Reflexionen, gegebenenfalls zur (teilweisen) Auslöschung des Signals.

Ein passiver Splitter (bestehend aus drei Widerständen) vermeidet zwar die Fehlanpassung, bringt aber eine zusätzliche Dämpfung mit sich, die man selten brauchen kann. Ein HF-Trafo als Verteiler ist eine andere Möglichkeit.

Erste Versuche

Zur Trennung könnte man aber auch ein Tiefpass- und ein Hochpassfilter mit passendem Frequenzgang kombinieren.

Um das auszuprobieren, habe ich zuerst zwei käufliche Filter durchgemessen. **Bild 2** zeigt die Frequenzgänge der Filter, jeweils getrennt mit einem Netzwerkanalyser vermessen. Die Frequenzen passen nicht zur UKW-DAB+-Trennung, aber beide Filter sind im Durchlassbereich schön flach.
Wenn man beide Filter per T-Stück parallel schaltet, entsteht eine starke gegenseitige Beeinflussung der Frequenzgänge (**Bild 3**). Danach habe ich noch mit einer Aufteilung durch Widerstands-Power-Splitter und zusätzlichen 6-dB-Dämpfungsgliedern nach dem Power-Splitter experimentiert. Dabei nimmt die gegenseitige Beeinflussung zwar ab, aber schön ist anders. Und es gibt eine zusätzliche Dämpfung von 3 bzw. 6 dB. Auch zeigt der Tiefpass einen abfallenden Frequenzgang. Das ist also keine optimale Lösung.

Diplexer-Software

Ein Diplexer kann hier die Lösung sein. Er stellt eine Kombination aus Tiefpass- und Hochpassfilter (oder Bandpass/Bandsperre) dar. Also eine Frequenzweiche für Hochfrequenz.
Beide Filter sind so aufeinander abgestimmt, dass sie parallel an einer Quelle betrieben werden (müssen). Zur Berechnung gibt es die Software „DiplexerDesign“ von James L. Tonne [1]. Bei diesem Programm kann zwischen der Lowpass/Highpass- und der Bandpass/Bandstop-Variante gewählt werden. Nachfolgend wird nur die Lowpass/Highpass-Variante beschrieben. Auf der „Design“-Seite dieses Programms (**Bild 4**) kann man die Eigenschaften des Diplexers festlegen. Die Ordnung ist ein Maß für den Aufwand (die Anzahl der Bauteile) und die damit erreichbare Steilheit des Filters. Die „Crossover freq“ gibt den Schnittpunkt beider Filter an. Für das Beispiel UKW/DAB+ ist es in etwa die Mitte zwischen höchster UKW-Frequenz (108 MHz) und tiefster DAB+ Frequenz (174 MHz). Die Übergangsfrequenz kann so fein abgestimmt werden, dass die Dämpfung der Filter beim jeweils unerwünschten Band in etwa gleich ist. Man kann auch versuchen, die krummen Spulenwerte auf verfügbare Werte hinzuschieben.
Der Tiefpass im oberen Teil beginnt immer mit einer Spule in Längsrichtung. Bei der Berechnung eines einzelnen Filters kann man wählen, ob man mit einer Spule in Längsrichtung (T-Struktur) oder einem Kondensator nach Masse (Pi-Struktur) startet. Dies ist hier nicht möglich. Entsprechend startet der Hochpass immer mit einem Kondensator in Längsrichtung.
Der „Passband ripple“ gibt die Welligkeit der Amplitude im Durchlassbereich an, ähnlich wie bei einem Tschebyscheff-Filter. Eine größere Welligkeit ergibt eine höhere Steilheit der Filter und umgekehrt.
„System Z“ gibt den Wellenwiderstand an, in unserem Beispiel 50 Ohm.
Drückt man den „Plot“-Button, so gelangt man zum nächsten Bildschirm (**Bild 5**).
Auch hier kann man die Ordnung, den Ripple und die Übergangsfrequenz verändern und direkt die Auswirkungen im Diagramm beobachten. Die Darstellung kann man unter „Plot Options“ einstellen, unter „Markers“ kann man Marker einblenden.
Der so errechnete Diplexer wurde zunächst in Simetrix simuliert und mit einem Tschebyscheff-Tiefpass verglichen.

Praktische Schaltung

Danach entwickelte ich eine Platine mit EAGLE. Die „krummen“ Werte der Spulen und Kondensatoren wurden durch Reihen- und

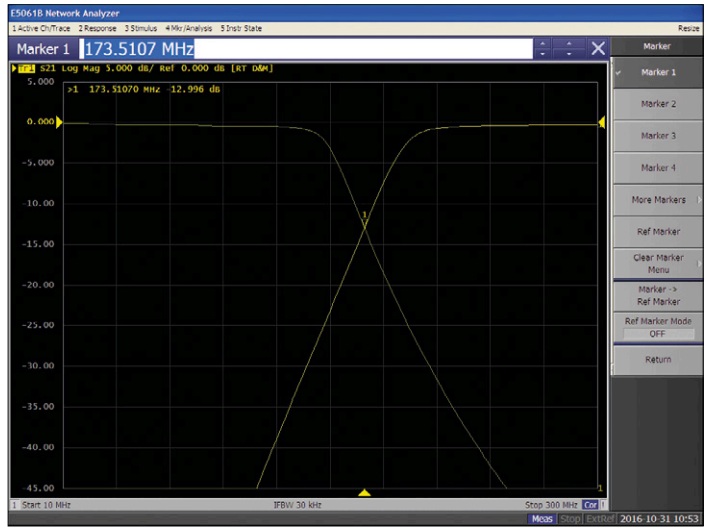


Bild 2. Tiefpassfilter SLP-150+ und Hochpass SHP-250+ von Mini-Circuits; jeweils getrennt gemessen mit einem Netzwerkanalysator.

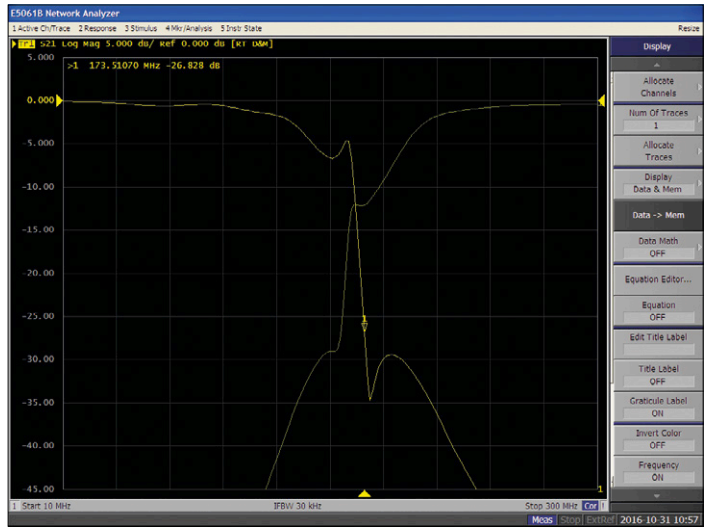


Bild 3. Beide Filter sind hier per T-Stück parallelgeschaltet. Dies erzeugt eine starke gegenseitige Beeinflussung der Frequenzgänge.

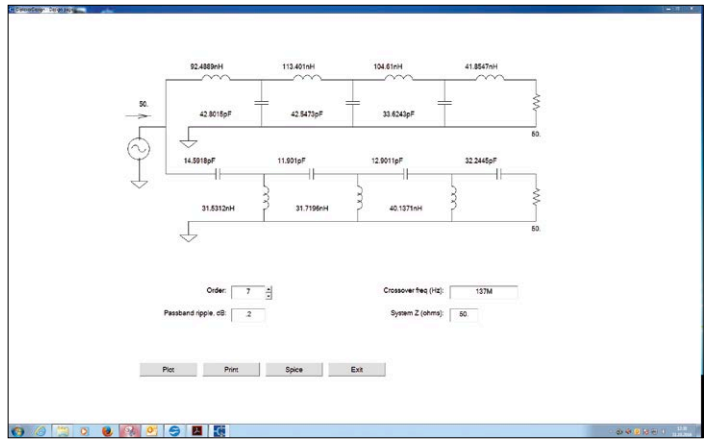


Bild 4. Design-Seite des Diplexer-Design-Programms.

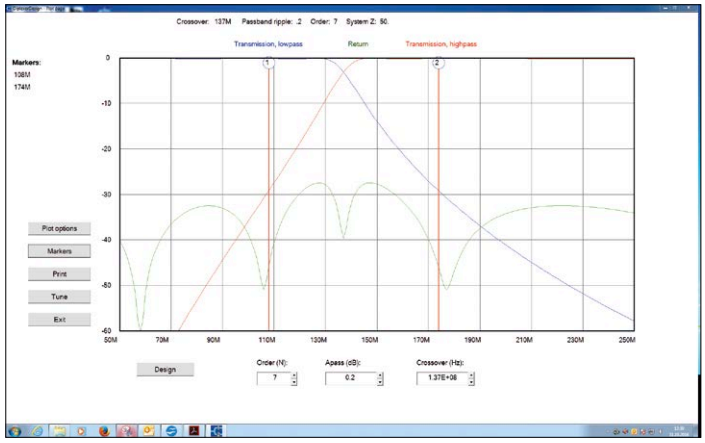


Bild 5. Plot-Seite des Diplexer-Design-Programms.

Parallelschalten erreicht. Es ist problemlos möglich, Kondensatoren parallelzuschalten und sie auch direkt nebeneinander anzuordnen. Anders ist es mit der Reihenschaltung der Spulen. Beeinflussen sich die beiden Magnetfelder, so entsteht ein (wenn auch sehr schlechter) Transformator und der effektive Spulenwert ist nicht die gewünschte Addition beider Werte. Daher sind die Spulen um 90 Grad versetzt und, soweit das bei einem SMD-Layout geht, voneinander entfernt platziert.

Die maximale Ordnung ist 7. Durch Weglassen von Spulen (die gebrückt werden müssen) und Kondensatoren kann man aber auch jede tiefere Ordnung realisieren. Die Spulen sind im 0603-Gehäuse erhältlich.

Der Schaltplan der praktischen Schaltung ist in **Bild 6** zu sehen, die zugehörige Platine in **Bild 7**. Diese Platine passt in ein kleines Plastikgehäuse und kann dort festgeschraubt werden (**Bild 8**). Die Kabel der Ein- und Ausgänge werden direkt auf die Platine gelötet. Zwischen den beiden Filtern gibt es einen Massesteg auf der Oberseite, die Unterseite ist komplett Masse. Die Messergebnisse in **Bild 9** bestätigen das Design im Rahmen der Bauteiltoleranzen. Bei ca. 45 dB Dämpfung geht der Tiefpass in die „Sättigung“. Dies ist zwar unschön, aber für die Funktion nicht erheblich.

Aktive Version

Da die Sendeleistung der DAB+-Sender im Vergleich zu den UKW-Sendern eher bescheiden ist, suchte ich nach einem geeigneten Verstärker und wurde beim MAX 2630 von Maxim fündig. Wie im Datenblatt [2] zu sehen ist, besitzt er ein 4-poliges SAT143-Gehäuse. Als Beschaltung braucht er neben dem Bypass-Kondensator an der Versorgungsspannung nur je einen Kondensator am Ein- und Ausgang. Da er wunderbar auf eine Lochrasterplatine passt, wurde zur Erprobung keine eigene Platine entwickelt. Seine Verstärkung beträgt rund 15 dB, die

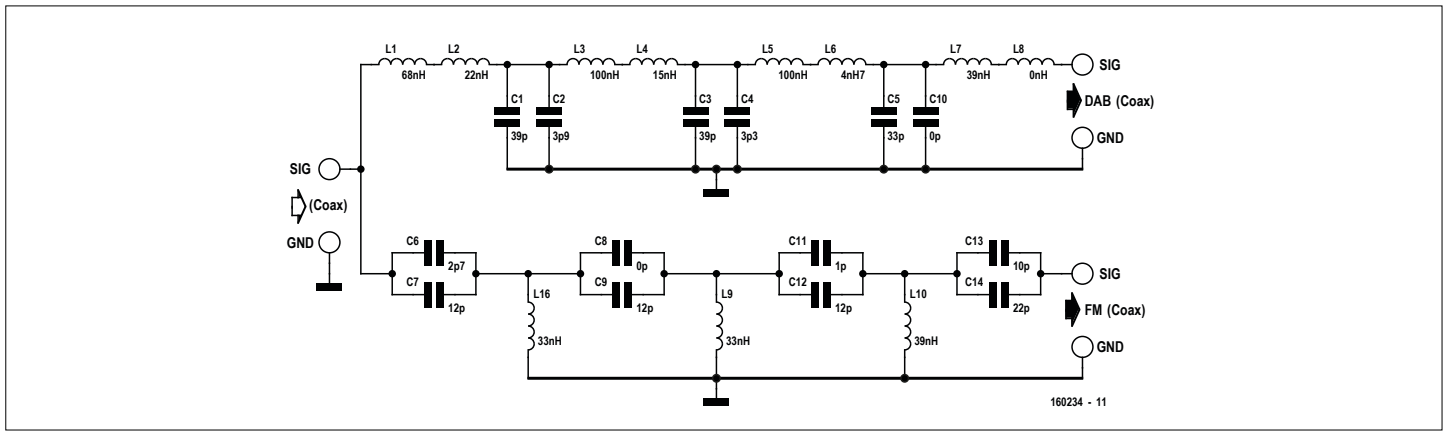


Bild 6. Schaltplan der passiven „Frequenzweiche“ für FM/DAB+.

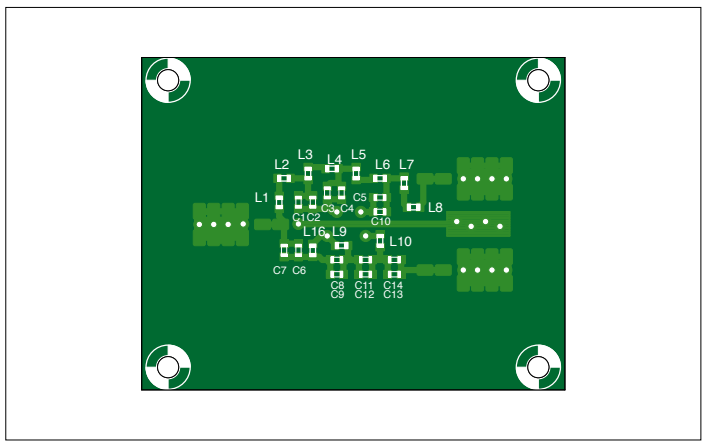


Bild 7. EAGLE-Layout für die Schaltung von Bild 6.

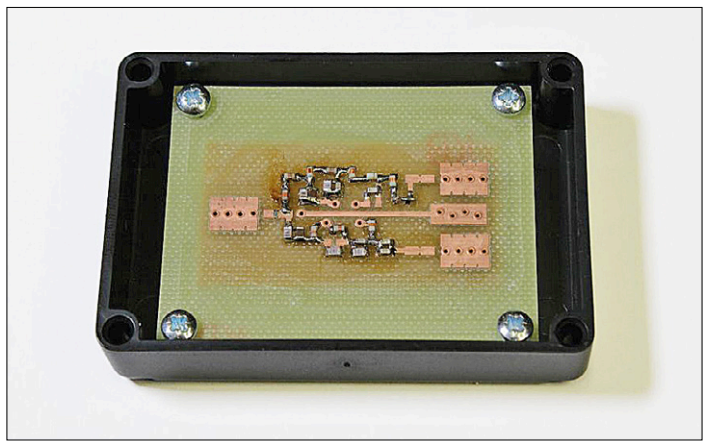


Bild 8. Der Prototyp des Diplexers im Gehäuse.

Rauschzahl liegt bei 4. Der gemessene Frequenzgang zeigt eine fast gerade Linie bis 250 MHz (und er geht deutlich weiter). Durch die Messergebisse ermutigt entwickelte ich einen aktiven Diplexer, siehe **Bild 10**. Sowohl am Eingang als auch an beiden Ausgängen ist jeweils ein MAX 2630 vorhanden.

Der Verstärker am Eingang macht vermutlich nur Sinn, falls im Auto eine passive Antenne verbaut ist. Für den höheren Pegel einer aktiven Antenne könnte es des Guten zu viel sein. Dann kann man den Ein- und Ausgang des Verstärkers mit einem Draht oder einem SMD-0-Ohm-Widerstand brücken. Ebenso am UKW-Ausgang, falls es zur Übersteuerung des Empfängers kommt. Der Verstärker am DAB+-Ausgang macht aber vermutlich meistens Sinn.

Die Stromversorgung erfolgt durch einen 3,3-V-Spannungsregler, ein Verstärker zieht etwa 7 mA. Die Längsdiode am Eingang verhindert eine Falschpolung, die beiden Widerstände sollten die thermische Verlustleistung des Reglers vermindern. Die 12 V können an einem Ausgang des Autoradios abgegriffen werden, der zur Versorgung einer aktiven Antenne verfügbar ist. Die Platine des aktiven Diplexers (**Bild 11**) passt in ein Alu-Druckguss-Gehäuse von Hammond (1550WQ).

(160234)

Weblinks

[1] www.tonnesoftware.com/diplexer.html

[2] https://datasheets.maximintegrated.com/en/ds/MAX2630-MAX2633.pdf

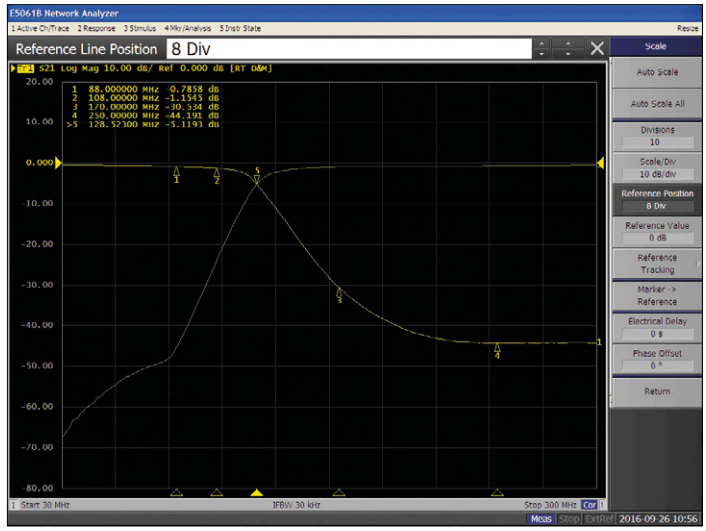


Bild 9. Frequenzgänge des Diplexers mit Markern.

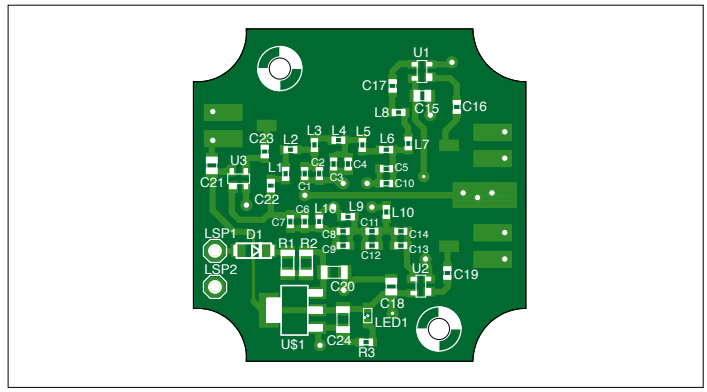


Bild 11. Layout der Platine für den aktiven Diplexer.

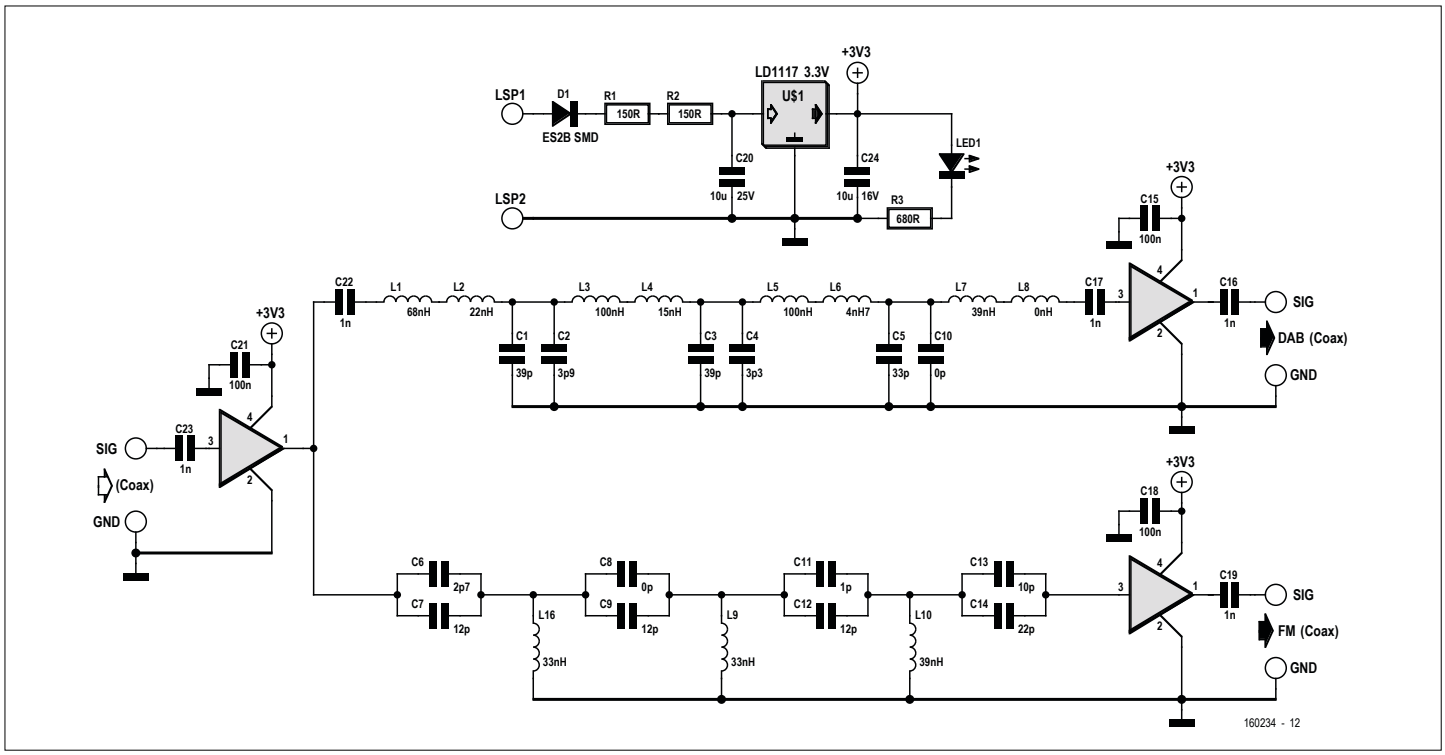


Bild 10. Schaltplan des aktiven Diplexers.

Connected Objects mit Genuino 101

Elektronik „funkt“ mit Mobilgerät

Von **Clemens Valens** (Elektor-Labor)

Um das Internet of Things und Connected Objects gibt es nun schon seit Jahren einen Hype. Industrie-Experten sprechen von Abermilliarden an verbundenen Geräten in den nächsten Jahren. Sie gehen von einem Marktvolumen im Bereich von zig Milliarden Dollar aus. Eindrucksvolle Zahlen! Wäre es nicht nett, wenn man davon etwas abhaben könnte? Ein erster Schritt dazu: Ein „Connected Object“.

Wenn es um „Connected Devices“ und IoT geht, denkt man üblicherweise an Geräte, die Daten per Funk von und zur Cloud schicken. Die Verbindung bedient sich dabei allerlei bekannter Techniken, doch hauptsächlich wird auf WLAN und Bluetooth gesetzt. Bluetooth 4.0 Low Energy bzw. BLE eignet sich dabei sehr gut für die Anbindung von (Ultra-)Low-Power-Geräten wie Sensoren oder Wearables an ein leistungsfähigeres Mobilgerät wie Smartphone oder Tablet, das eine Art Brücke zur Cloud darstellt.

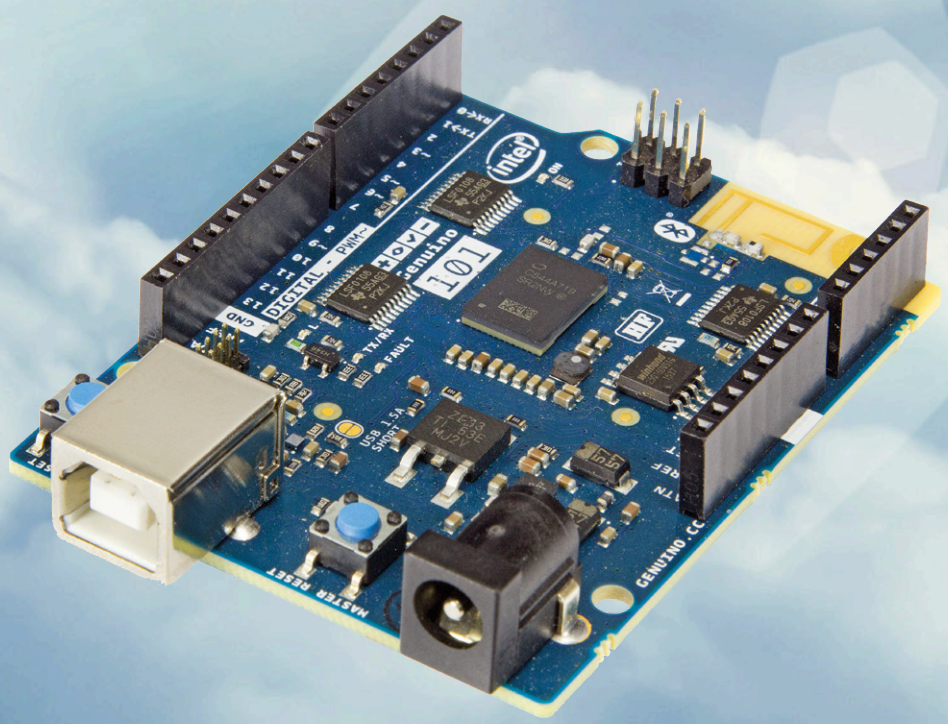
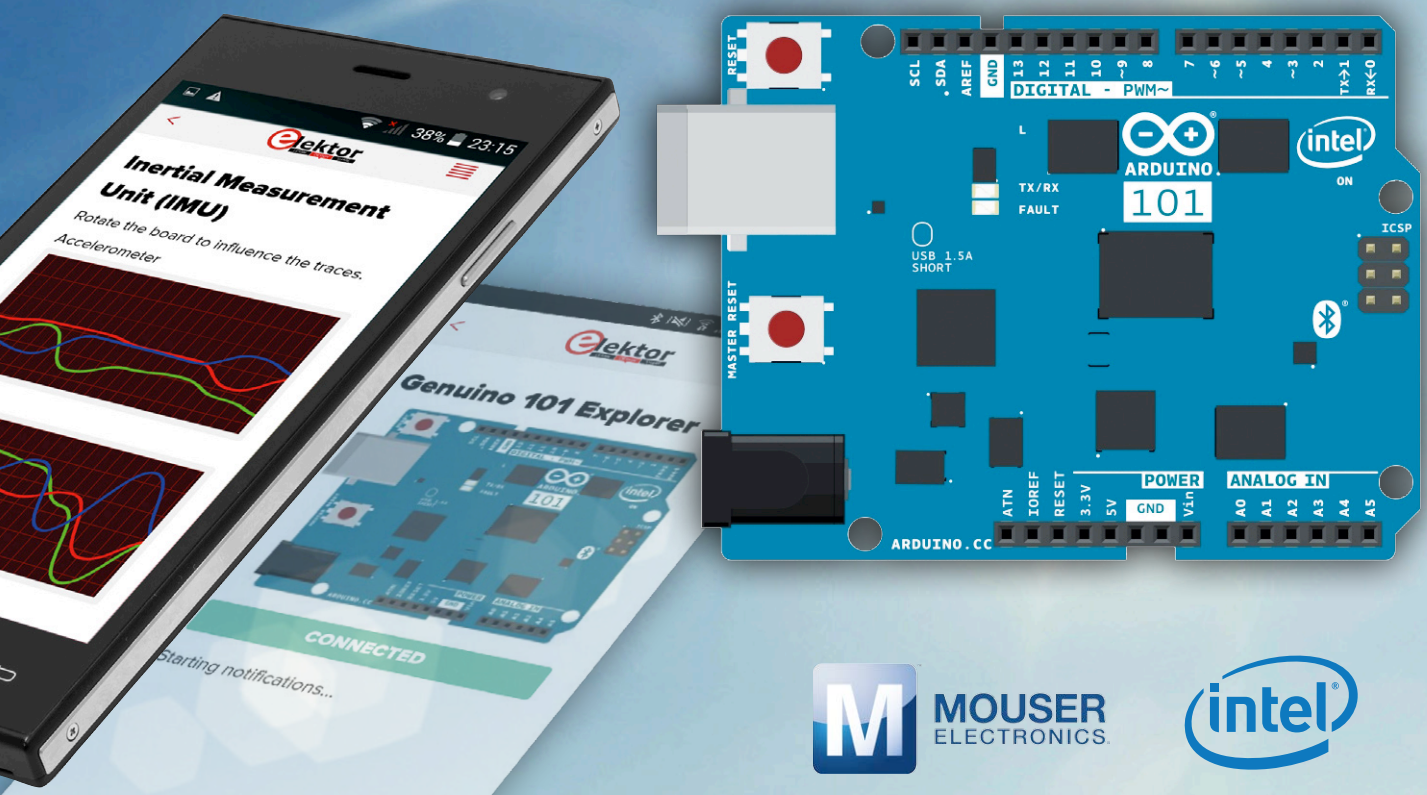


Bild 1. Das Intel Arduino/Genuino-101-Board verfügt über eine sechsachsige IMU und BLE.

Beim Genuino 101 (Arduino 101 in den USA) von Intel (**Bild 1**) handelt es sich um ein Arduino-kompatibles Board, das voll von den verbreiteten und kostenlosen Arduino-Entwicklungs-Tools unterstützt wird. Dank BLE ist es eine ideale Plattform für die Entwicklung eigener „Connected Objects“. Die Rechenpower liefert ein Curie-Modul von Intel mit seinem integrierten 32-bit-Quark-Mikrocontroller. Geboten werden 384 KB Flash, 80 KB SRAM, BLE, eine sechsachsige Gyroskop-Accelerometer-Kombination und eine Ladeschaltung für Akkus. Das Board ist zu Arduino-Shields bzw. dessen Steckverbindungen voll kompatibel. Es stehen 20 digitale I/Os und sechs analoge Eingänge zur Verfügung

In diesem Beitrag geht es um die Anbindung eines „101“ an ein mit BLE ausgestattetes Mobilgerät unter Android 4.4 (KitKat) oder neuer. Offiziell wird BLE ja seit Android 4.3 unterstützt, doch dank mangelnder Zuverlässigkeit sollte man



sich das nicht antun.
Zuerst wird ein Erweiterungs-Board (Arduinesisch: „Shield“) mit BME280-Sensor zur Erfassung von Temperatur, Luftdruck und Luftfeuchte eingesetzt [1]. Darüber hinaus werden Daten der 101-IMU (Inertial Measurement Unit, bzw. Gyrometer und Accelerometer) erfasst und als gefällige Grafik auf dem Mobilgerät angezeigt [2]. Natürlich kann mal damit auch Werte der Analogeingänge sehen und die digitalen Ausgänge steuern.
Zur Verbindung von Genuino 101 per BLE mit einem Mobilgerät braucht es eine App. Da das „Connected Object“ maßgeschneidert ist, muss man auch die passende App selbst erstellen. Viele Wege führen zur Android-App, doch einige sind steiniger als andere. Anfänger bevorzugen einfachere Methoden. Ein prima Tool ist z.B. *Evothings*. Sein Konzept ist einfach: Es wird ein Basis-App-Framework zur Verfügung gestellt, zu der man eigenen Code mit JavaScript, HTML und CSS hinzufügen kann – ganz wie aus der Webseitenerstellung bekannt. Tatsächlich mutiert die Programmierung einer App zur Erstellung einer kleinen Webseite, die dann im Viewer angezeigt wird und so auf dem Mobilgerät läuft. Aus diesem Grund kann die App auch jederzeit sehr einfach geändert werden, ganz ohne langwierige Compiler-Läufe. Man muss lediglich etwas Code schreiben bzw. ändern. Sobald dieser gesichert ist, wird Evothings den Viewer damit updaten. Das Ergebnis kann man sofort auf seinem Mobilgerät sehen. Das ist super praktisch, nicht wahr? Kein Crash auf dem Mobilgerät etc., denn der Viewer fängt Fehler ab. Wenn man dann mit dem Resultat zufrieden ist, kann man eine „richtige App“ erzeugen und sogar via Google Play Store öffentlich anbieten.

Das klingt nicht schlecht, oder? Also in die Hände gespuckt...

Voraussetzungen

- Genuino 101
- BME280-Shield
- PC (Windows, Linux oder OSX) mit einem freien USB-Port, ein Text-Editor und optional ein Programm zur Bildbearbeitung (mein Setup: Windows 7, SP1, 64-bit)
- Mobilgerät mit BLE unter Android 4.4 oder neuer (mein Handy: Samsung Galaxy J5 mit Android 5.1.1)
- Internetzugang für PC und Mobilgerät

Hardware-Zusammenstellung

Zur Fertigstellung der Hardware muss man eigentlich nur das BME280-Shield auf das 101-Board stecken (**Bild 2**). Wer das weitertreiben will, kann mit einem eigenen Erweiterungs-Board mit einigen Widerständen, LEDs und Trimmern etwas mit den digitalen und analogen I/Os spielen, aber das ist nicht zwingend. Falls man das aber macht, sollte man Pin-Konflikte mit dem BME280-Shield vermeiden. Die digitalen Pins 0...6 und die Analogeingänge 0...3 sind frei.

Software-Installation

Man muss recht viel Software installieren und konfigurieren, bevor man loslegen kann. Zuerst geht es einfach los:

- PC: Download und Installation der Arduino-IDE (www.arduino.cc, meine Version: 1.6.10)
- PC: Download und Installation von Evothings Studio (http://evothings.com, meine Version: 2.0.0)
- Mobilgerät: Download und Installation des Evothings Viewer (Google Play Store, meine Version: 1.4.1)

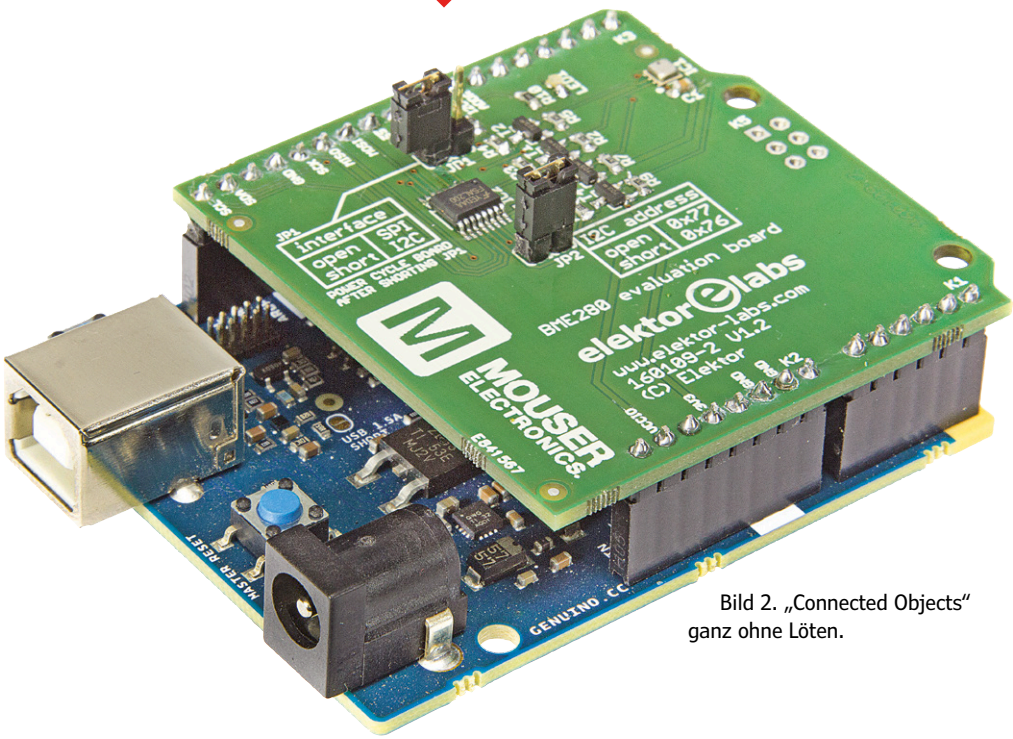


Bild 2. „Connected Objects“ ganz ohne Löten.

Genuino 101

Das Board wird noch nicht an den PC angeschlossen. Zuerst müssen einige Treiber installiert werden. Man startet die Arduino-IDE und öffnet den Boards Manager (Tools -> Boards -> Boards Manager). In der oberen linken Ecke wählt man *Arduino Certified*, was eine kürzere Liste ergibt, aus der man Intels *Curie* auswählt. Dann ein Klick auf den Install-Button. Abhängig von Ihrer Internetverbindung kann die Installation etwas dauern. Anschließend schließt man den Manager und geht wieder in das Menü Tools -> Boards, scrollt dort bis zum Eintrag *Arduino/Genuino 101* und wählt diesen aus.

Jetzt kommt das Board an den PC. Es sollte ein serieller Port verfügbar sein. Dieser wird in der Arduino-IDE (Tools -> Port) ausgewählt.

Wenn die Installation des 101 Schwierigkeiten macht, empfiehlt sich die Lektüre des Online-Installation-Guide: www.arduino.cc/en/Guide/Arduino101.

Es folgt der Download des BME280-Sketch von [1]. Der QR-Code auf der Rückseite des BME280-Shield enthält die passende URL (http://bit.ly/2aNNDq7). Jetzt lädt man den Sketch auf das 101-Board, wartet einige Sekunden und öffnet den *Arduino IDE Serial Monitor* (Tools -> Serial Monitor). Kurz darauf sollte man die Werte für Temperatur, Luftdruck und Luftfeuchte sehen können. Bevor es weitergeht, sollte das Shield funktionieren. Jetzt kommt noch der Download des *Genuino 101 Explorer Sketch* von [1]. Er wird auf das 101-Board geladen. Der Serial Monitor wird sich nach einigen Sekunden zeigen - die Daten indizieren, dass das Board auf eine Bluetooth-Verbindung wartet (**Bild 3**).

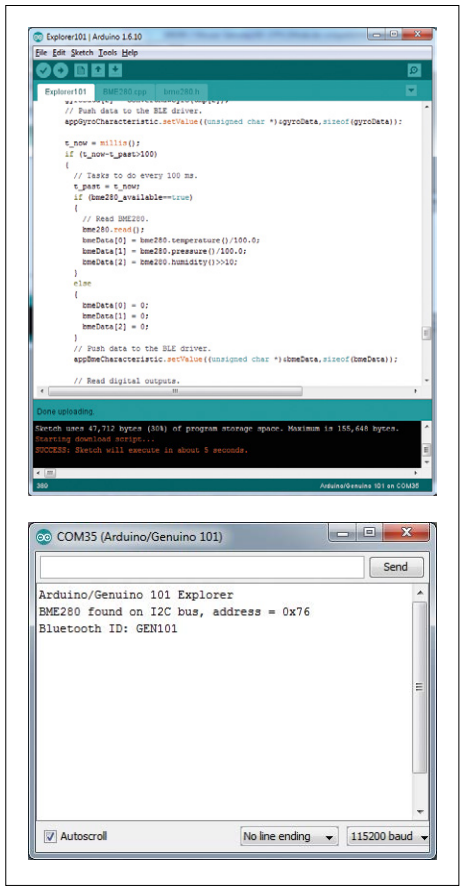


Bild 3. Erfolgreich hochgeladener Sketch. Das Board wartet jetzt auf eine Bluetooth-Verbindung.

Evothings

Evothings Studio wird auch *Evothings Workbench* genannt. Zur Installation startet man seine ausführbare Datei. Im Tab *New* gibt man *Genuino-101-Explorer* als Verzeichnisname der App ein. Bevor man aber auf *Create* klickt, sollte man sich das Verzeichnis notieren, in der die App gesichert wird. Dann wird Evothings Workbench wieder geschlossen.

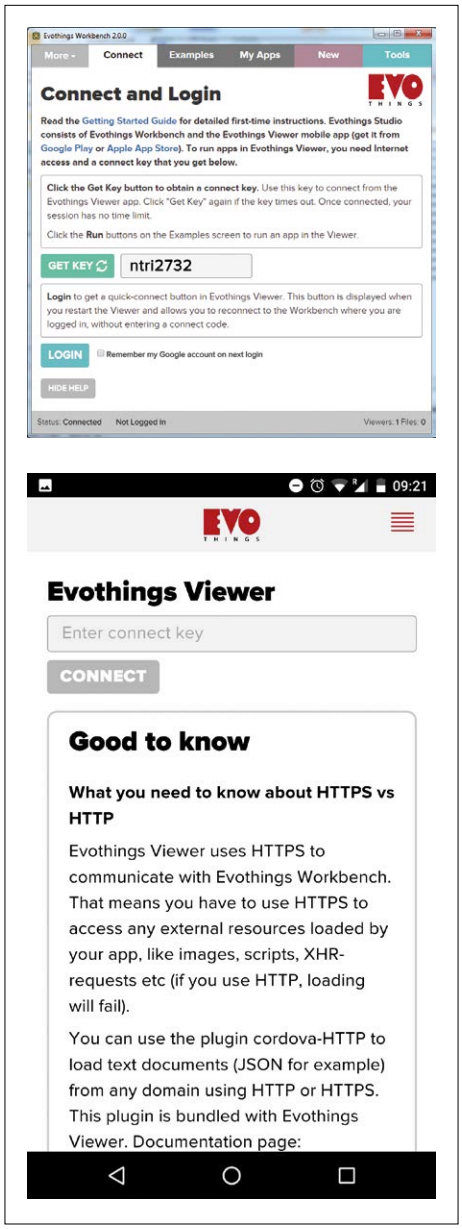
Mit einem Datei-Manager navigiert man zum gerade angelegten App-Verzeichnis und löscht dessen Inhalt. Von [1] lädt man jetzt *Genuino 101 Explorer Evothings Project* herunter und entpackt das Archiv da hinein. Jetzt sollten einige Verzeichnisse und Dateien, darunter *index.htm* und *app.js* zu sehen sein. Diese beiden Dateien sind wichtig. Nun Evothings Workbench und den Tab *MyApps* öffnen. Dort sollte ein Eintrag namens *Arduino/Genuino 101 Explorer* existieren.

Jetzt auf dem Mobilgerät die Evothings-Viewer-App starten (**Bild 4**). Nach dem Öffnen verlangt sie nach einem *Connect Key*. Dieser Schlüssel wird von Evothings Workbench erzeugt: Ein Klick auf den Tab *Connect* und dann auf *Get key*. Nun gibt man den Schlüssel im Viewer ein und tappt auf *Connect*. Damit dies klappt, müssen PC und Mobilgerät mit dem Internet verbunden sein. Bei erfolgreicher Verbindung informiert der Viewer über die nächsten Aktionen. Man muss dann den Workbench-Tab *MyApps* öffnen und auf den Button *Run* der App *Arduino/Genuino 101 Explorer* klicken. Anschließend erscheint ein Loading-Icon im Viewer, und einige Sekunden später öffnet die App.

Wenn die Internet-Verbindungen von Mobilgerät und PC gut und stabil sind, bleibt die Verbindung zwischen Viewer und Workbench bestehen. Andernfalls darf man bizarre Fehlermeldungen bewundern, und man versucht besser eine neue Verbindung mit neuem Schlüssel.

Ausprobieren

Wenn Ihr Genuino 101 nicht an den PC angeschlossen ist, sollten Sie das jetzt tun. Einschalten nicht vergessen. Bei schon bestehender Verbindung genügt ein Druck auf den Reset-Taster (nicht den Master-Reset-Taster). Nach einigen zig Sekunden Pause checkt man mit dem Serial-Monitor, ob das Board auf eine Verbindung wartet. Dann tappt man auf den roten Button *Connect* des Evothings-Viewer auf dem Mobilgerät. Wenn darauf



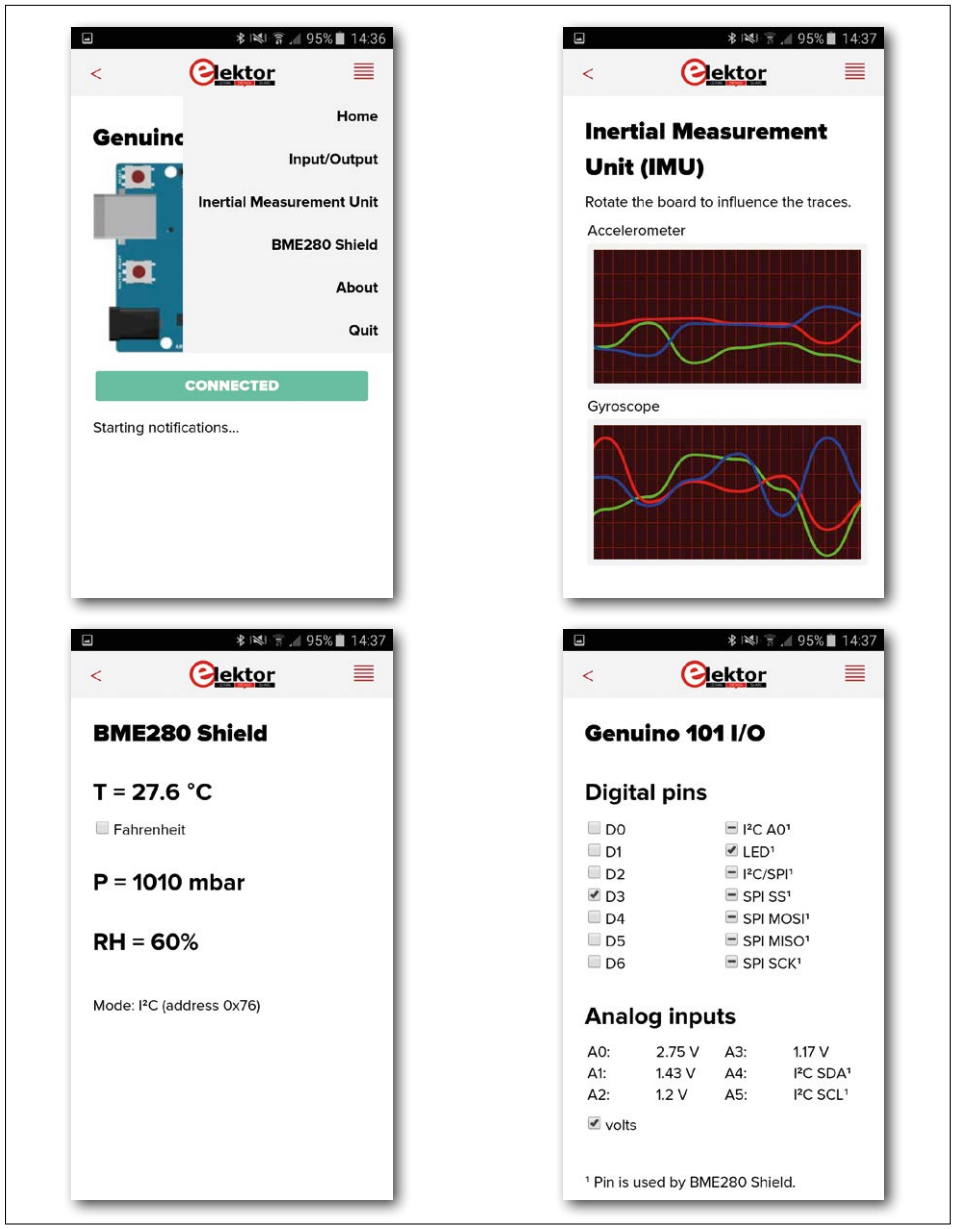


Bild 4. Evothings Workbench auf dem PC und der Viewer auf dem Mobilgerät.

Bild 5. Screenshots der App mit unterschiedlichen Datenansichten.

Bluetooth noch nicht aktiviert war, wird jetzt zwecks Aktivierung nach einer Autorisierung gefragt. Hier stimmt man zu und beobachtet die Meldungen unter dem Button *Connect*. Wenn alles klappt, wird der Button nach einigen Sekunden grün. Nun ein Tap auf den Button *Menu* in der oberen rechten Ecke, und man kann eine Datenansicht auswählen (**Bild 5**).

Zum Sketch

Nachdem das funktioniert, wird es Zeit für weitere Systemkomponenten. Zunächst zum Sketch für Genuino 101. Es basiert auf *CurieBLE*- und *CurieIMU*-Beispielen (Arduino IDE -> File -> Examples) zu denen ich den BME280-Treiber hinzugefügt habe. Da hier der Schwerpunkt auf BLE liegt, wird das Hardware-Interfacing übergangen.

BLE nutzt Dienste, die sich durch einzigartige IDs identifizieren. Einige Dienste wurden von der Bluetooth SIG (www.bluetooth.com) definiert, doch da unsere Wetter/IMU-Kombination (noch) nicht existiert, braucht es hierfür eine eigene ID. Hierfür erzeugt man (z.B. mit einem Online UUID-Generator) die nötige UUID (Universal Unique Identifier). Nun muss noch für jede Art der übertragenen Daten eine sogenannte Charakteristik zum Service hinzugefügt werden, die wieder eine eigene ID braucht. Als Charakteristiken kommen *read-only*, *write-only* oder *read-write* in Frage - diese generieren u.U. Benachrichtigungen.

In der Theorie kann eine Charakteristik bis zu 512 Bytes übermitteln, aber die BLE-Library von Genuino 101 ist auf 20 Bytes begrenzt. Unsere Applikation produziert drei 4-Byte-Fließkomma-Werte für das Accelerometer (= 12 Bytes) und ebenfalls 12 Bytes für das Gyroskop und den BME280. Die analogen Werte passen in je 2 Bytes, was wiederum 12 Bytes ergibt. Die digitalen I/Os benötigen nochmals 2 Bytes. Insgesamt würden also 50 Bytes reichen. Um die Sache zu vereinfachen, wird für jeden Datentyp eine Charakteristik erstellt, um sie individuell zu übertragen.

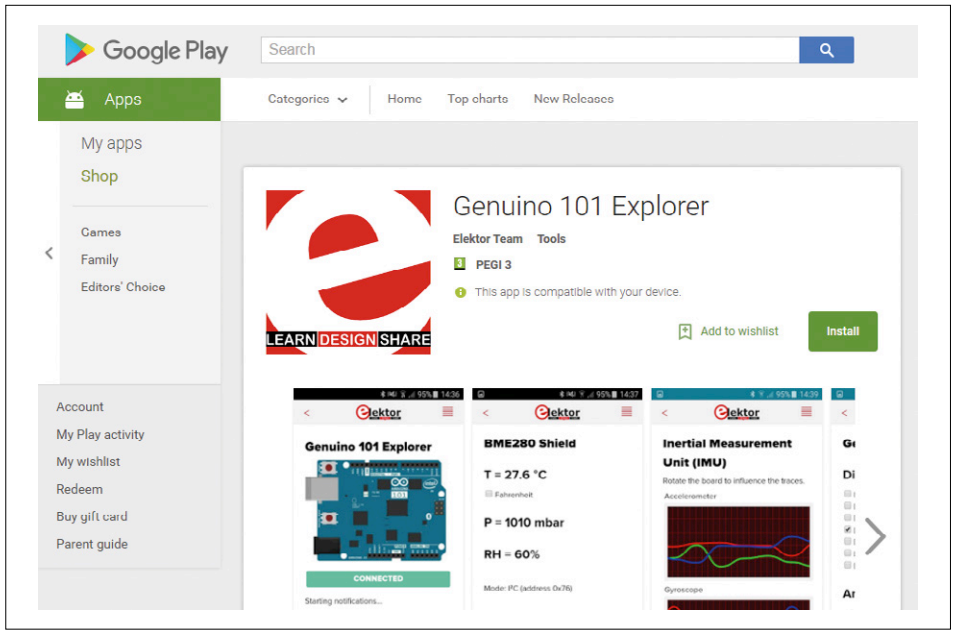


Bild 6. Die eigene App im Google Play Store – ab sofort ist man im IoT-Geschäft!

Wenn die Charakteristiken definiert, initialisiert und als Attribute zum Service hinzugefügt sind, kann die BLE-Peripherie starten. Ohne Verbindung passiert erstmal nichts, aber wenn diese besteht, kann man schon Daten mit diesen Charakteristika senden und empfangen. Die wirkliche Arbeit erledigt die Library für uns. IMU-Daten werden so schnell wie möglich zum Mobilgerät gestreamt - die anderen Daten lediglich alle 100 ms.

Das Einlesen von Daten einer Charakteristik funktioniert auf zwei Arten: Durch Polling per Funktion *written()* oder eventgesteuert durch Registrierung eines *Listeners*. Beide Verfahren werden für die digitale I/O-Charakteristik im Sketch Genuino 101 Explorer illustriert.

Dem Service wird ein (lokaler) Name verpasst: *„GEN101“*. Man sollte diesen kurzhalten, denn er muss in den geringen Platz des *Advertising Buffers* passen. Wenn der Name nicht passt, erzeugt die Library einen eigenen, wodurch die Fähigkeit zur automatischen Erkennung der App gestört wird.

Über die App

Die App-Seite ist ganz ähnlich: Der eigentliche Code steckt in der JavaScript-Datei *app.js*. Das UI (User Interface) besteht aus der HTML-Datei *index.htm*, und das Style-Sheet *ui\css\evothings-app.css* bestimmt, wie die HTML-Elemente aussehen.

Wie Genuino 101 verfügt auch Evothings über eine nette BLE-Library, die sich um das ganze Bluetooth-Gedöns kümmert. Die App muss natürlich auch die für den Genuino-101-Sketch definierten UUIDs kennen. Da die Charakteristika asynchron einlaufen, aktiviert die App Empfangsbenachrichtigungen, um die Daten so entgegen zu nehmen, wie sie hereinkommen. Die Daten für analoge und digitale I/Os werden sofort nach Ankunft angezeigt. Die restlichen Daten werden vor ihrer Anzeige erst gespeichert. Für Grafik und BME280 Daten gibt es Intervall-Timer. Die Grafiken werden langsam erneuert, sodass Board-Vibrationen und Rauschen schön ausgemittelt werden und die Verläufe „smooth“ aussehen. Die Glättung ergibt sich schon aus der verwendeten Grafik-Library: *Smoothie Charts* (http://smoothiecharts.org).

Die Kommunikation zwischen UI (HTML) und der App (JavaScript) ist bidirektional. Von *app.js* zu *index.htm* wird ausgiebig von der Funktion *getElementById()* Gebrauch gemacht. Damit dies klappt, muss das zu verändernde HTML-Element eine ID haben. Die Property *innerHTML* überschreibt den Text innerhalb des Elements. HTML-Code ist erlaubt, wodurch die Seite dynamisch neu gezeichnet wird. Mit den Methoden *getAttribute* und *setAttribute* kann man Element-Eigenschaften abfragen sowie ändern und so z.B. deren Farbe oder Größe verändern.

Der HTML-Code kann ebenfalls Funktionen von *app.js* aufrufen, wenn z.B. ein Anwender auf Knöpfe oder Menüs klickt oder tappt oder einen Haken auf einer Auswahlbox ändert. Es wird eine JavaScript-Funktion für das *Onclick-Attribut* des Elements deklariert: z.B. *onclick="app.onStartButton()"*.

JavaScript- und HTML-Code verbinden sich über das Internet, wie z.B. bei der About-Seite der App, wo einige Basislinks dem Anwender das Öffnen von Webseiten erlauben. Mit etwas mehr Code erhält man auch Verbindung zu Online-Speicherdiensten oder Twitter und kann so Daten im Web veröffentlichen.

Die eigene App

Wenn man meint, dass die bisher erstellte App reif für die Verwandlung in eine richtige Android-App ist, muss man sie kompilieren. Sehr gut: Das klappt sowohl für Android als auch für IOS - eine Träne: leider nicht für Windows. Ein Compiler-Lauf ist einfach, wenn man über die Cordova-Toolchain von Apache verfügt (http://cordova.apache.org). Ihre Installation erfordert viel Software:

- Node.js (meine Version: 4.4.7)
- NPM (meine Version: 2.15.8)
- Git (meine Version: 2.6.3.windows.1)
- Cordova (meine Version: 6.3.1)
- Java JDK (meine Version: jdk1.8.0_101_x64)
- Android SDK (meine Versionen: 25.1.7 + API 24 + API 23 + API 19)

Glücklicherweise gibt es online hierfür recht detaillierte Anleitungen: http://evothings.com/doc/build/build-overview.html#Install, was die Beschreibung hier erspart.

Wenn Cordova funktionierend installiert ist, erzeugt man nach folgendem Rezept eine App für das Evothings-Projekt. Doch zunächst muss man ein Cordova-Projekt erstellen:

Erstellung eines Arbeitsverzeichnisses und Öffnen des CLI (Command Line Interface).

Im Verzeichnis ausführen von „`cordova create [my_project] [com.elektor.labs.my_app] [my_app_name]`“ – alles in den eckigen Klammern wird durch eigene Bezeichnungen ersetzt. Der Name mit den vielen Punkten entspricht einer URL mit umgekehrter Reihenfolge.

Im CLI folgt ein „`cd`“ zum erstellten Projektverzeichnis: „`cd [my_project]`“.

Alles im Unterverzeichnis *„www“* wird gelöscht, nicht aber das Unterverzeichnis selbst.

Kopieren des Evothings-Projekts in das Verzeichnis „*www*“ inkl. aller Dateien und Unterverzeichnisse mit *app.js* und *index.htm*.
Im CLI wird der Befehl „`cordova plugin add cordova-plugin-ble`“ ausgeführt, um die BLE-Library zum eigenen Projekt hinzuzufügen. Dies wiederholt man für andere benötigte Plugins.
Im CLI gibt man den Befehl „`cordova platform add android`“ ein. Für IOS lautet dies „`cordova platform add ios`“. Man kann auch mehr als eine Plattform hinzufügen.

Das Projekt erstellt man per CLI mit den Befehlen:

```
cordova build android
cordova build ios (not on Windows)
```

Normalerweis sollte ein „Build“ ohne Warnungen oder Fehlermeldungen klappen. Das Resultat findet sich anschließend in *platforms/android/build/outputs/apk*. Um das Ergebnis auszuprobieren muss man die APK-Datei (IPA für IOS) auf das Mobilgerät kopieren und mit dessen Dateimanager installieren. Dazu muss das Gerät die Installation aus unbekannter Quelle zulassen (siehe Sicherheitseinstellungen). Um eine App im Google Play Store zu veröffentlichen, muss sie nochmals im Release-Mode kompiliert und signiert werden. Hierzu muss man zuvor einen Schlüssel generieren (einzeiliger Befehl):

```
keytool -genkey -v -keystore [Name].
keystore -alias [Alias] -keyalg RSA
-keysize 2048 -validity 10000
```

Dabei wird `[Name]` und `[Alias]` durch Ihre Wahl ersetzt. Nun ergänzt man die verlangten Infos und notiert sie sich für später. Noch ein Lauf (einzeiliger Befehl; Vorsicht bei den doppelten Bindestrichen):

```
cordova build android --release
-- --keystore="[Path]\[Name].
keystore" --storePassword=[Password]
--alias=[Alias]
```

`[Name]` und `[Alias]` sind gleich wie zuvor, aber `[Path]` verweist auf die neue Keystore-Datei, und `[Password]` wurde bei der Key-Erzeugung schon angelegt. Ein kleines Pop-up-Fenster verlangt die Bestätigung des Passworts – anschließend noch ein Klick auf „*OK*“. Es wurde eine .APK-Datei mit der Bezeichnung „android-release“ erzeugt.
Wenn Sie noch keinen Publisher-Account für den Google Play Store haben (kostet einmalig 25 $), laden Sie die APK-Datei hoch, füllen die Formulare aus, veröffentlichen die App (**Bild 6**) und schon sind Sie im Geschäft!

Fallstricke

Was tun, wenn die Android-App nicht das Gleiche tut wie die im Viewer? Vermutlich liegt das daran, dass nicht alle Probleme gefixt wurden, die der Evothings-Viewer geduldig abgefangen hat. Die Bugs sucht man am besten mit der JavaScript-Konsole der Evothings Workbench im Tab *Tools*, und lässt die App dann im Viewer laufen. Alles was „Unreferenced“ und „Unhandled“ ist, muss gefixt werden.
Man sollte im Hinterkopf behalten, dass manche Timeouts für reale Situationen zu kurz sein können, und dass die Dinge sich anders verhalten, wenn das Mobilgerät und Objekt verbunden sind. Auch könnte der Bluetooth-Stack irgendwas blockieren. Vielleicht muss man auch die Verbindung beenden und neu aufbauen. Und wenn Ihre App unerwartet aussieht? Dann öffne man die Datei *index.htm* in einem Browser auf dem PC. Sie sollte so wie im Viewer ausschauen, nur größer. Wenn nicht muss man auch hier nach Fehlern suchen.
Noch ein Wort zu BLE. Das saubere Handling von BLE ist schwierig, da es fast nicht möglich ist, eine App komplett zu beenden. Android lässt sie nämlich einfach im Hintergrund laufen, wo sie noch teilweise eine Verbindung aufrechterhalten kann. Um dies zu vermeiden sollte man die App mit dem Applikation-Manager „abschießen“ und sicherheitshalber auch das verbundene Objekt neu starten.

Fazit

In diesem Artikel wurde gezeigt, wie man ein Genuino-101-Board mit einem BLE-fähigen Smartphone oder Tablet lediglich mit Hilfe von Open-Source und freien Tools koppelt. Das Board wurde hierzu mit einem BME280-Shield zur Erfassung von Temperatur, Luftdruck und Luftfeuchte ausgestattet. Nachdem ein eigener BLE-Service angelegt und einige Charakteristika definiert wurden, war der Datenaustausch zwischen den beiden verbundenen Objekten möglich. Dies ist typisch dafür, wie man IoT-Daten eines Accelerometers, Gyroskops oder Wetterdaten auf ein Mobilgerät holt. Eine native App stellt auf dem Mobilgerät ein geeignetes UI zwecks Datenanzeige und User-Interaktion zur Verfügung und sorgt für die Datenübertragung von und zur Cloud. Abschießend wurde die App auch noch im Google Play Store veröffentlicht, damit auch andere Zeitgenossen etwas davon haben.

(160109-2)

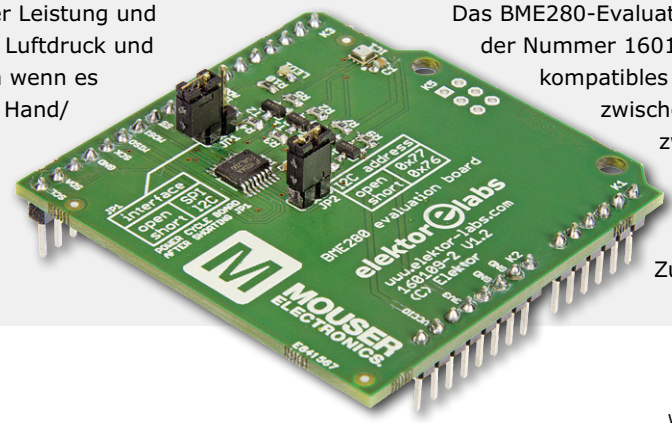
Weblinks

[1] www.elektormagazine.com/labs/bme280-evaluation-Board-160109-2

[2] www.hackster.io/gov/imu-to-you-ae53e1

BME280 Evaluation Board

Der Sensor BME280 von Bosch Sensortec ist ein Bauteil, das mit hoher Leistung und hochpräzise Temperatur, Luftdruck und Luftfeuchte erfasst. Auch wenn es eher auf Mobilgeräte zur Hand/Gesichtserkennung und für Indoor-Navigation abzielt, fungiert es doch prima auch als Wetterstation. Es kann per SPI oder per I²C angesprochen werden.
Das BME280-Evaluations-Board ist von Elektor unter der Nummer 160109-2 erhältlich. Es ist ein Arduino-kompatibles Shield mit SPI- oder I²C-Verbindung, zwischen denen man per Jumper wählt. Ein zweiter Jumper selektiert eine von zwei I²C-Adressen. Modus und I²C-Adresse können nicht nur abgefragt, sondern auch per Befehl gesetzt werden. Zusätzlich ist noch eine LED verfügbar.



Einfacher Temperaturwächter

Schaltet ein und aus

Schaltungen zur Überwachung von Temperaturen werden heutzutage sehr vielseitig verwendet: In Kühlschränken, Wasserboilern, Mikrowellenherden und vielen anderen Geräten werden dabei Sensoren wie zum Beispiel der TMP36 zusammen mit einem Mikrocontroller und der entsprechenden Programmierung als Thermostate eingesetzt. Doch es geht auch ohne Prozessor!

Von **Bera Somnath** (Indien)

Technische Daten TMP36

- Betriebsspannung: 2,7..5,5 Volt
- Arbeitstemperatur: –40..+150°C
- Linearer Bereich: $y = mx + c$
- Linearität: 20 mV/°C
- Konvertiert Temperaturen direkt in Millivolt
- Ruhestrom: < 50 µA
- Aufwärtsverhalten: 3°C/s
- Abwärtsverhalten: 6°C/s

Kaum ein Elektronik-Projekt, das heute ohne Mikrocontroller und die entsprechende Software auskommt. Doch es gibt immer noch Schaltungen, die auch ohne Prozessor zuverlässig arbeiten. Der kostengünstige Temperatur-Sensor TMP36 wird zwar häufig in Mikroprozessorschaltungen eingesetzt, deren Steuersignale dann an daran angeschlossene Relais oder andere Bauelemente weitergeleitet werden, doch beim hier beschriebenen Projekt funktioniert das Ganze auch ohne dazwischengeschalteten Mikrocontroller: Mit nur ein paar diskreten Bauteilen steuert der Sensor das angeschlossene Relais auf einfache und präzise Art und Weise.

Schaltung

Bild 1 zeigt das Symbol und die Anschlussbelegung des TMP36. Die gemessenen Temperaturen werden direkt als Spannungen in Millivolt ausgegeben und an das Sziklai-Transistorpaar [1] T1 und T2 weitergeleitet. Übersteigt die Spannung an der Basis von T1 die Ruhestromgrenze, so wird T2 leitend und das mit 5 V betriebene Relais wird aktiviert. Die Sziklai-Schaltung hat gegenüber der Darlington-Variante den Vorteil, dass hier die Basis-Emitter-Spannung nur 0,61 V statt 1,22 V beträgt und so auch die Verarbeitung sehr niedriger Spannungen möglich ist.

Das Projekt kann auf kleinstem Raum aufgebaut und mit einer 9-V-Blockbatterie bzw. einer maximalen Spannung von 12 V versorgt werden. Der Standard-Spannungsregler 7805 sorgt für eine stabilisierte, interne Versorgungsspannung von 5 V. Dank des relativ geringen Stromverbrauchs kommt man mit einer Batterie eine relativ lange Zeit aus.

Abgleich

Durch den Wegfall eines Mikrocontrollers bleibt der Abgleich sehr einfach. Mit dem Spindelpoti P1 wird die temperaturabhängige Schaltschwelle eingestellt. Heizen Sie den Sensor bis zur gewünschten Schalttemperatur auf und stellen Sie P1 so ein, dass das Relais aktiviert wird. Steigt die Temperatur über diesen Punkt hinaus an, so bleibt das Relais eingeschaltet. Sinkt sie unter den betreffenden Wert, so muss sich das Relais wieder ausschalten. Je nach Anwendung kann natürlich auch der mit NC bezeichnete, invertierte Kontakt des Relais verwendet werden.

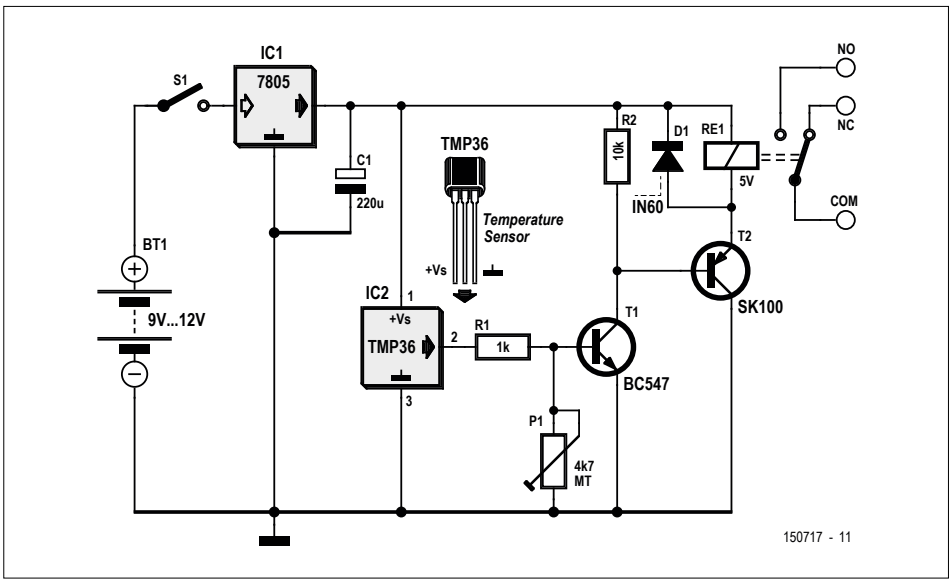


Bild 1. Eine einfache Schaltung mit wenigen Bauteilen.

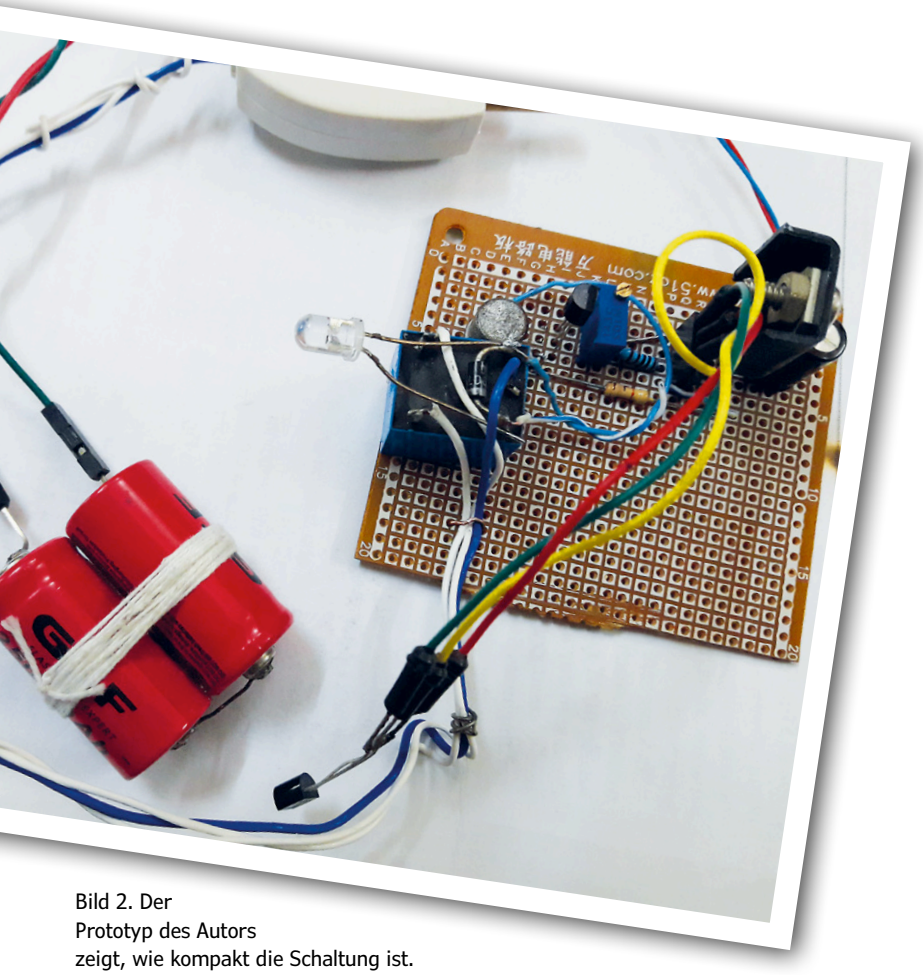


Bild 2. Der Prototyp des Autors zeigt, wie kompakt die Schaltung ist.

Schwingverhalten

Zur Vermeidung von Schwingungen um eine Durchschnittstemperatur besitzt der TMP36 ein unterschiedliches Verhalten beim Ansteigen und Abfallen der Umgebungstemperatur: Er folgt mit 3 °C/s, wenn sie steigt, und mit 6 °C/s, wenn sie sinkt. Außerdem hat der Sensor durch seine physikalische Masse bereits ein gewisses Temperatur-Beharrungsvermögen, das schnelle Oszillationen verhindert.

Anwendungsmöglichkeiten

Die Schaltung kann zum Beispiel zur Überwachung der Akkutemperatur beim Laden verwendet werden, um ein Überladen zu verhindern. Der Ladevorgang sollte beim Erreichen einer bestimmten Temperatur (etwa 40 bis 45 °C bei Ni-MH-Akkus und 40 °C bei Lithium-Ionen- und LiPo-Akkus) beendet werden. Die Akkus werden es Ihnen durch eine längere Lebensdauer danken.

Auch zur Messung der Temperatur an der Spitze geregelter Lötkolben eignet sich der hier vorgestellte Temperatursensor recht gut. Dazu sollte sich der Sensor nicht zu nahe an der Spitze befinden, sondern in einem Bereich, in welchem die Temperatur den Wert von 150 °C nicht übersteigt. Zum Ausschalten des Lötkolbens wird in diesem Falle der (invertierende) NC-Anschluss des Relais verwendet. ◀

(150717)

Weblinks

[1] https://de.wikipedia.org/wiki/Sziklai-Paar

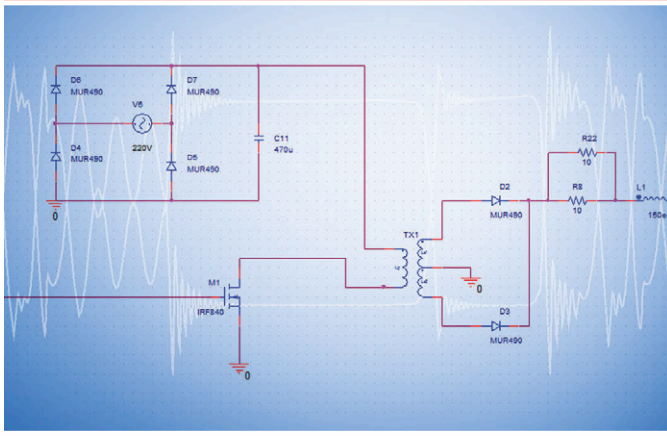
[2] www.elektormagazine.com/150717

Anzeige

FlowCAD

Elektronik einfach erklärt

100+ kostenlose Video-Tutorials mit PSpice-Schaltungen zum selbst simulieren



FlowCAD stellt auf YouTube mehr als 100 Tutorials zu Grundschaltungen der Elektronik zur Verfügung.

In den Videos werden die elektronischen Schaltungen schrittweise und verständlich erklärt. Die Funktion der Schaltungen wird mit PSpice simuliert.

Alle Schaltungen stehen zum kostenlosen Download zur Verfügung und können so selbst schnell mit der PSpice LITE Version nachvollzogen werden.

Mehr Informationen unter:

Youtube.de/FlowCAD

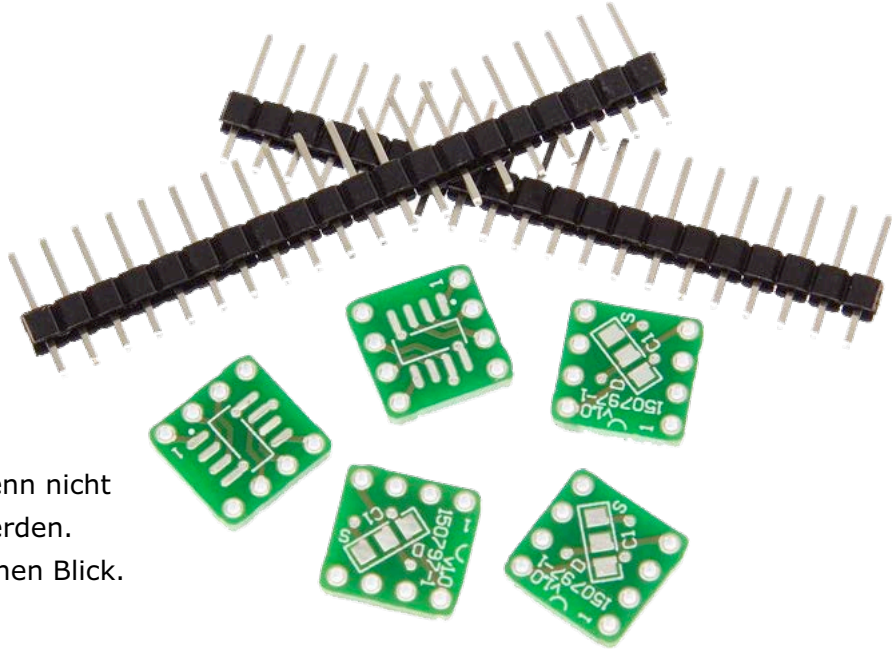
oder

www.FlowCAD.de/uni

PSpice cādence

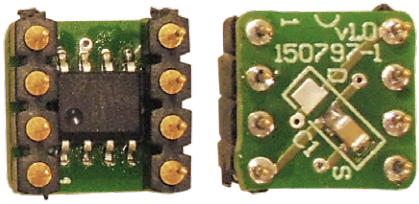
Klein, aber fein...
Elektor BoBs

Von **Jelle Aarnoudse** (NL)

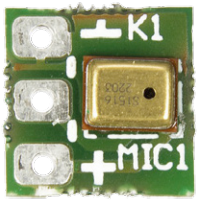


Das Entwickeln kleiner, aber feiner Break-out-Boards (BoBs) ist bei uns fast schon Tradition geworden. Trotz der hohen Beliebtheit bei unseren Lesern haben wir „den Kleinen“ nur selten ausführliche Beschreibungen gewidmet, denn nicht immer müssen viele Worte geschrieben werden. Hier finden Sie die wichtigsten BoBs auf einen Blick.

SO8 nach DIP8 Adapter (150797)



Den Auftakt macht ein ebenso simples wie kleines BoB. Die winzige Platine, etwa so groß wie das Viertel einer Briefmarke, stellt eine Brücke zwischen SMD-Chips der Bauform SO8 und Platinen her, die für das Anschlusslayout DIP-8 ausgelegt sind. Der SMD-Chip hat seinen Platz auf der Unterseite zwischen den Stiftkontakten. Auf der Oberseite kann ein Entkoppelkondensator montiert werden, zwischen Pin 4 (meistens Masse) und Pin 7 (zum Beispiel bei Opamps) oder Pin 8 (bei vielen anderen ICs). Als Satz zu fünf Stück im Elektor-Shop erhältlich.

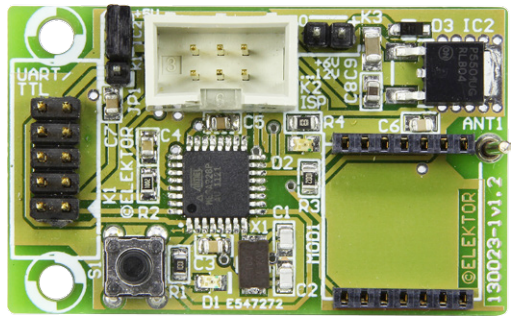


MEMS-Mikrofon Adapter (150346)

Doch es geht noch kleiner. Den Fledermaus-Detektor[PLUS] hatten wir mit einem MEMS-Mikrofon ausgestattet, das so winzig und schwierig zu löten war, dass ein Adapter mehr als sinnvoll erschien. Auch das vollständige Modul gehört zum Angebot des Elektor-Shops. Übrigens: Wegen der außergewöhnlichen Eigenschaften des MEMS-Mikrofons ist dieses BoB längst nicht nur für Fledermaustöne geeignet.

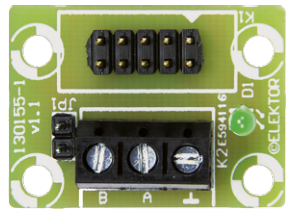
Drahtlos-Gateway (130023)

Ein Entwicklungstrend, der sich in der Elektronik schon länger abzeichnet, ist die drahtlose Signal- und Informationsübertragung. Neben das HMI (Human-Machine Interface) tritt das MMI (Machine-Machine Interface) im IoT (Internet of Things). Deshalb besteht dringend Bedarf an unkomplizierten und trotzdem sicheren drahtlosen Übertragungssystemen. Die Industrie entwickelt mit Hochdruck Chips und Module für diesen Zweck. Die zugehörigen „Datenblätter“ haben allerdings den Charakter dickleibiger Bücher, die Anwendung ist wenig gebrauchsfreundlich. Deshalb hat das Elektor-Labor einige BoBs mit 433-MHz-Transceiver, ATmega328 und UART entwickelt. Ausgestattet mit vorprogrammierter Firmware lassen sich diese BoBs unkompliziert als drahtlose Gateways einsetzen. Kommando- und Datenstrings werden vom UART auf Funk umgesetzt – und umgekehrt. Lieferbar als einsatzbereite Module vom Elektor-Shop.



RS485-Modul für ECC (130155)

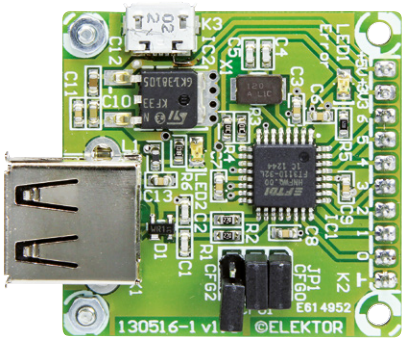
Auch für weniger komplexe Arten der Kommunikation ist das Prinzip „Anschließen und Loslegen“ meistens hoch willkommen. So entstand das RS485-Modul für den ECC (Embedded Extension Connector). Damit kann ein Mikrocontroller-Board durch eine RS485-Schnitt-



stelle ergänzt werden, sie ist für zuverlässige Kommunikation, auch über weite Abstände, und für Buskonfigurationen geeignet. Gibt es fix und fertig im Elektor-Shop.

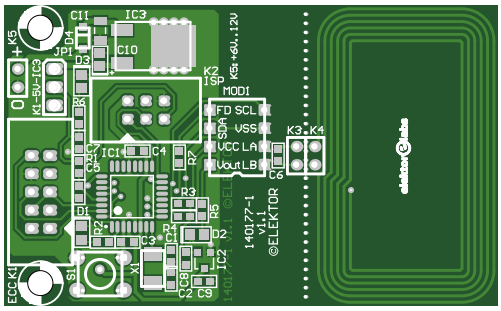
FT311D BoB (130516)

Zum Steuern und Testen selbst entwickelter Boards wird fast immer eine Eingangs- und Ausgangsschnittstelle benötigt. Ein Smartphone, das heute fast jeder besitzt, ist dazu bestens geeignet. Der Chip FT311D wurde entwickelt, um Android-Geräte über USB mit externer Elektronik zu verbinden. Zwar glänzt USB nicht gerade durch Unkompliziertheit, doch dieser Chip besitzt außer einer USB-Host-Schnittstelle diverse Anschlüsse zum Steuern externer Elektronik, beim FT311D sind es sieben. Außerdem hat der Hersteller in diesen Chip bereits die Firmware geladen, die nach dem AOAM-Protokoll die vollständige USB-Kommunikation abwickelt. Der Elektor-Shop hat sowohl die leere Platine als auch das fertige Modul im Sortiment.



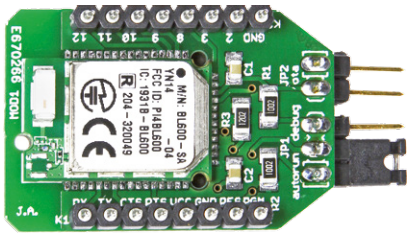
NFC Gateway (140177)

Auch dieses BoB dient der Kommunikation. Viele Mobiltelefone sind heute mit NFC ausgestattet. Damit können über Abstände bis zu einigen Zentimetern Daten ausgetauscht werden, beispielweise bei Bezahlvorgängen an Kassen. Für Elektroniker ist dies natürlich hochinteressant: Wie funktioniert das, können wir einen „NFC-Tag“ auslesen oder sogar beschreiben? Ja, das geht, aber einfach ist es nicht. Deshalb haben wir für unsere Leser ein NFC-Gateway entwickelt, auf dem ein Mikrocontroller ATmega328 die Hauptarbeit übernimmt. Außerdem sind ein NFC-Tag und eine integrierte Antenne vorhanden. Weil sich der NFC-Tag-Chip nur äußerst mühsam löten lässt, liefern wir ihn auf einem BoB, das Gateway ist dagegen als unbestückte Platine lieferbar.



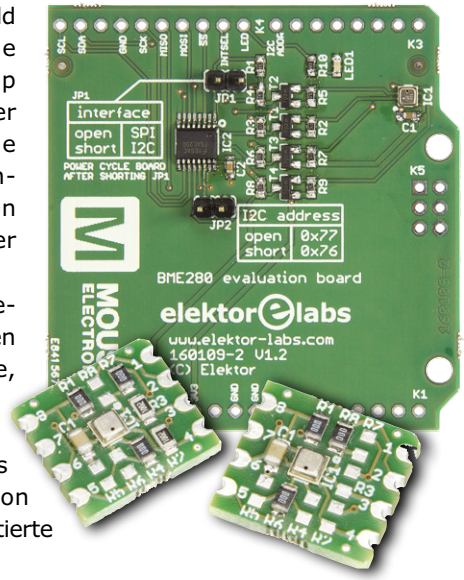
BL600 e-BoB Bluetooth Low Energy Modul (140270)

Noch ein drahtloses Modul! Das e-BoB BL600 ist eine für Steckbretter ausgelegte Schnittstellenplatine, die eine ultra-sparsame Funkkommunikation möglich macht. Auch für den Langzeiteinsatz reicht eine Knopfzelle wie die CR2032 aus. Im Elektor-Shop als fertiges Modul erhältlich.



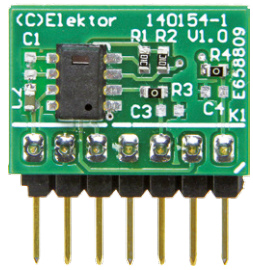
BME280 Arduino-Shield (160109)

Der BME280 ist ein Sensor von Bosch, der Temperaturen, Luftfeuchten und Drücke messen kann. Damit lässt sich beispielsweise eine Ein-Chip-Wetterstation realisieren. Mit diesem vom Elektor-Labor entwickelten Arduino-Shield ist der erweiterte Zugang zu dem Chip möglich, sowohl über SPI als auch I²C. Die integrierte Pegelanpassung erlaubt den Einsatz in 3,3-V- oder 5-V-Umgebungen.
Der Elektor-Shop bietet mehrere Versionen an: Die leere Platine, die leere Platine für das Entwicklungsboard, eine bereits montierte I²C-Version und eine bereits montierte SPI-Version.



CC2A-eBoB Temperatur- und Feuchtesensor (140154)

Dieser präzise, vielseitige Sensor weist eine Genauigkeit von 2 % auf, er wird individuell kalibriert und getestet. Trotz SMD-Bauform lässt er sich noch gut von Hand löten. Natürlich passt er nicht auf ein Steckbrett, dafür ist er zu klein. Deshalb haben wir ihn auf eine Mini-Platine mit Stiftkontakten gesetzt.



Elektor Uno R4 (150790)

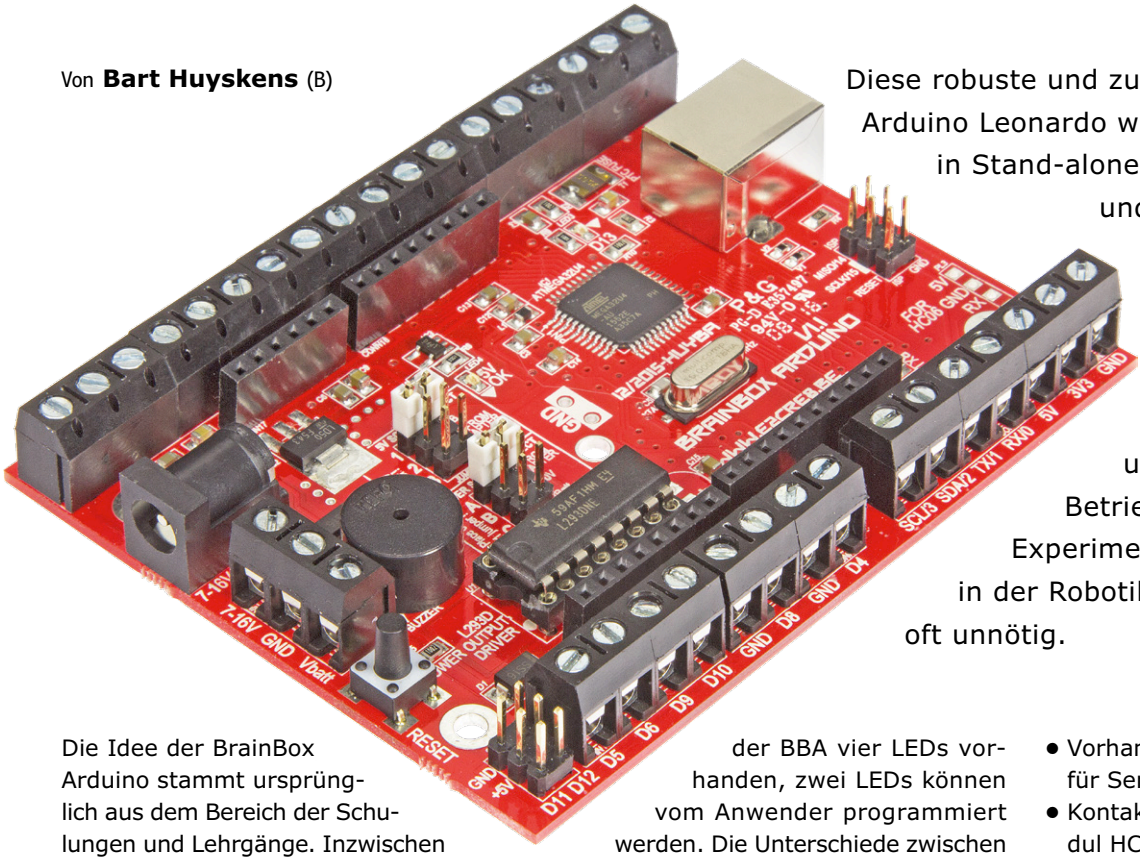
Zum Schluss darf in dieser Übersicht das aus dem Elektor-Labor stammende Board Uno R4 nicht fehlen, eine Weiterentwicklung des verbreiteten Arduino Uno R3. Das Board Uno R4 bietet umfangreichere Konnektivität, es ist flexibler und robuster. Eine geeignete Plattform für IoT-Entwicklungen! ◀

(150797)gd

BrainBox Arduino

Solider Arduino mit Kabelschraubklemmen

Von **Bart Huyskens** (B)



Diese robuste und zuverlässige Version des Arduino Leonardo wurde für den Einsatz in Stand-alone-Projekten entwickelt und ist auch für Lehr- und Lernzwecke prädestiniert. Der Onboard-Motortreiber-Baustein, die Kabelschraubklemmen und die wählbare Betriebsspannung machen Experimente einfach, zum Beispiel in der Robotik. Externe Hardware ist oft unnötig.

Die Idee der BrainBox Arduino stammt ursprünglich aus dem Bereich der Schulungen und Lehrgänge. Inzwischen wurde das Board so weiterentwickelt, dass sich mit ihm unzählige interessante Projekte realisieren lassen. Das Programmieren der BrainBox Arduino, nachfolgendkurz BBA genannt, ist ohne weitreichende Vorkenntnisse und ohne viel Zeitaufwand in mehreren Programmiersprachen möglich. Einige Demos wurden parallel unter fünf verschiedenen Entwicklungsumgebungen erstellt.
Im Elektor-Shop ist die aufgebaute BBA mit vorprogrammiertem Bootloader lieferbar, so dass Löterfahrung nicht zwingend vorausgesetzt wird. Ferner ist ein Lernpaket für den Bau des abgebildeten Robotkart im Angebot, das die Hardware, die Software und die Courseware umfasst. Gesteuert wird das Robotkart über eine individuell zu schreibende App.

Extras

Kernstück der BBA ist der leistungsstarke Mikroprozessor des Leonardo, ein ATmega32U4, getaktet mit 16 MHz. Dieser Mikroprozessor ist über die USB-Schnittstelle programmierbar. Wie beim Arduino Leonardo sind auch bei der BBA vier LEDs vorhanden, zwei LEDs können vom Anwender programmiert werden. Die Unterschiede zwischen den beiden Varianten betreffen insbesondere die erweiterte Funktionalität, die von der BBA verglichen mit den Standard-Arduino-Boards mitgebracht wird:

- Ein Tongeber, er kann Signale unterschiedlicher Frequenzen wiedergeben, zum Beispiel beim Testen individuelle Alarmtöne.
- Eine Doppel-H-Treiberbrücke, die vier Leistungsausgänge steuert, jeder Ausgang ist bis 600 mA belastbar. Entweder können vier DC-Motoren in Halbbrückenschaltung oder zwei DC-Motoren in Vollbrückenschaltung gesteuert werden. Die Leistungsausgänge sind auch für den Anschluss von Power-LEDs, Heizwiderständen oder Schrittmotoren geeignet. Die Spannungsversorgung (4,5…36 V) der Leistungsausgänge ist über die Spannung 5 V, ein Netzteil, einen Akku oder eine externe Versorgung möglich.
- Die Betriebsspannung des Mikroprozessors ist zwischen USB, Netzteil oder Akku mit Jumpern wählbar.
- Vorhanden sind zwei Steckverbinder für Servomotoren.
- Kontakte für ein Bluetooth-Modul HC06, für RS232 sowie I²C sind ebenfalls vorhanden.

Die Extras ermöglichen die Realisierung zahlreicher Projekte, ohne dass Hilfsschaltungen auf Steckbrettern oder zusätzliche Arduino-Shields notwendig wären.

Hardware

Die Schaltung der BBA in **Bild 1** mag auf den ersten Blick kompliziert erscheinen. Anschaulicher, insbesondere für Lehr- und Lernzwecke, ist die Zeichnung in **Bild 3**, aus der die Belegung der einzelnen Schraubklemmen hervorgeht. Dort sind auch die wichtigsten Komponenten zusammen mit kurzen Beschreibungen dargestellt.

Kernstück der BBA ist der Mikrocontroller ATmega32U4, ein Low-power 8-bit-RISC-Typ mit 32 KB Flash-Speicher, 2,5 KB SRAM und 1 KB EEPROM. Auf dem Chip sind eine USB 2.0 Full-speed-Schnittstelle, ein 12-kanaliger 10-bit A/D-Wandler und eine JTAG-Schnittstelle für ein On-chip-Debugging integriert. Auf dem BBA wird der ATmega32U4 von Quarz

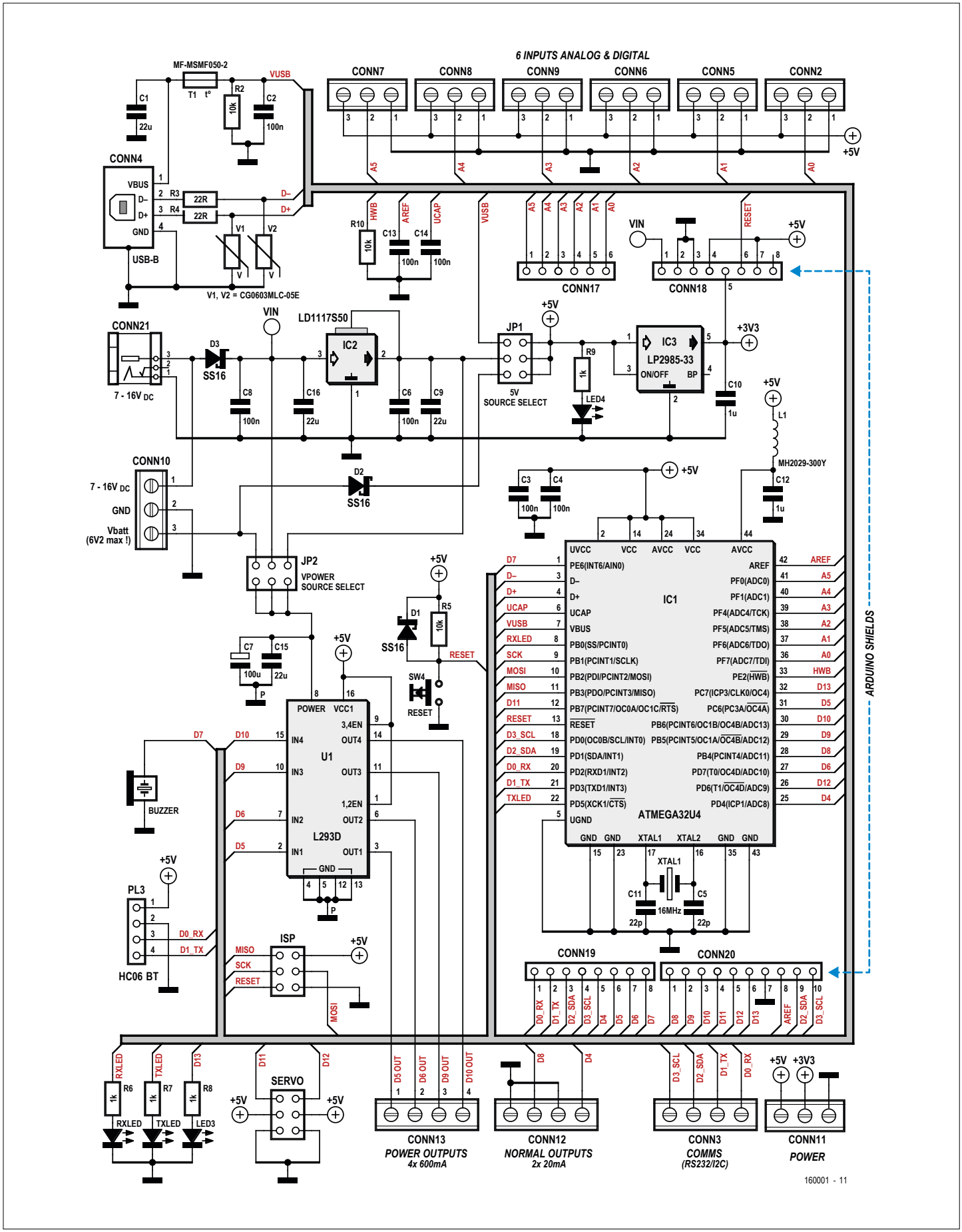


Bild 1. Hardware der BrainBox Arduino, der Mikrocontroller ist ein ATmega32U4.

Die BrainBox Arduino macht das Entwickeln und Programmieren komplexer Roboter ohne vertiefte Vorkenntnisse möglich.

XTAL1 mit seiner maximalen Frequenz 16 MHz getaktet. USB-Anschluss CONN4 ist über die Schutzwiderstände R3 und R4 unmittelbar mit dem Mikrocontroller verbunden. Der Anschluss der Betriebsspannung liegt über die rückstellbare Sicherung T1 an JUMPER1. Mit ihm kann die Betriebsspannungsquelle des Board gewählt werden.
Im Motortreiber U1, einem L293D, ist eine doppelte H-Brücke integriert, die vier Leistungsausgänge (CONN13) steuert. Jeder Ausgang ist bis 600 mA belastbar. Die Steuerung des Motortreiber-Bausteins erhält über Spannungsregler IC2 ihre Betriebsspannung. Für die Leistungstreiber im Baustein ist ein separater Anschluss vorhanden, hier kann die Spannungsquelle mit JUMPER2 gewählt werden.
Zwei der vier LEDs dienen zur Kontrolle der seriellen Kommunikation, die über Anschluss CONN3 stattfindet. Die RXLED kann auch eine andere Funktion übernehmen. Die dritte LED signalisiert das Vorhandensein der Betriebsspannung, die vierte LED (LED3) ist frei verfügbar. Mit Anschluss 1 des ATmega32U4 ist ein Tongeber verbunden, und an Anschluss 13 liegt der Reset-Taster.

Der Stromversorgungsteil der BBA ist vergleichsweise umfangreich, damit unterschiedliche Energiequellen wahlweise genutzt werden können. An CONN21 kann ein Netzteil mit einer Spannung 7…16 V angeschlossen werden. Alternativ ist ein Netzteil oder ein Akku-Pack mit maximal 6,2 V an die dreipoligen Schraubklemmen CONN10 anschließbar. Die Spannung 7…16 V gelangt über Verpolsicherung D3 zum 5-V-Spannungsregler IC2, er versorgt den Mikrocontroller und den Motortreiber U1 mit Spannung. Daran schließt sich ein 3,3-V-Spannungsregler (IC3) an, die Spannung 3,3 V liegt an den Shield-Steckverbindern und an Schraubklemme CONN11.
Die Wahl der Spannungsquelle wird mit den Jumpern JUMPER1 und JUMPER2 vorgenommen. JUMPER1 ist für die Spannungsquelle des Boards zuständig, USB-Anschluss, Netzteil oder Akku-Pack stehen zur Wahl. Mit JUMPER2 wird die Spannungsquelle der Motoren ausgewählt, die mit Motortreiber U1 verbunden sind. Hier sind die stabilisierte Spannung 5 V von IC2, das Netzteil oder der Akku-Pack durchschaltbar.

Der Rest der Schaltung besteht aus Steckverbindern und Kabelschraubklemmen. Für ein hinzugefügtes Arduino-Shield sind CONN17…20 bestimmt. Ferner sind sechs dreipolige Schraubklemmen-Gruppen mit analogen und digitalen Eingängen

Software-Optionen

Außer der IDE des Arduino sind weitere Entwicklungsumgebungen verfügbar, in denen die BrainBox Arduino programmiert werden kann. Eine nützliche und sinnvolle Beigabe zur BBA sind die Programmbeispiele, sie wurden für die fünf nachstehend aufgeführten Entwicklungsumgebungen erstellt. Der Anwender kann die BBA in der Umgebung programmieren, mit der er am besten vertraut ist.

Arduino-IDE mit Bibliotheken
Die originale Arduino-IDE ist ein unkomplizierter C-Compiler, zu dem zahlreiche Bibliotheken gehören. Die Bibliotheken und die Anschlussbezeichnungen, die dem Arduino eigen sind, machen einerseits das kurzfristige Schreiben komplexer Programme ohne vertiefte Mikrocontroller-Kenntnisse möglich. Andererseits lernt der Anwender nur wenig über die Technik und den Umgang mit Mikrocontrollern.

Arduino-IDE ohne Bibliotheken
Wenig bekannt ist die Tatsache, dass die Arduino-IDE auch mit realen Registernamen und Anschlussbezeichnungen zurechtkommt. Der Anwender muss sich allerdings die Mühe machen, den Mikrocontroller kennenzulernen, bevor er gebrauchsfähige Programme schreiben kann. Auch Kombinationen beider Varianten, mit und ohne Bibliotheken, sind möglich.

Flowcode (7) für AVR
Aus didaktischer Sicht ist Flowcode zweifellos der beste Weg, in das Embedded Programming einzusteigen. Die Flowcharts veranschaulichen visuell die Programmierstrukturen, und mit dem Simulator lässt sich der Programmcode zuverlässig testen, bevor er in die Hardware geladen wird. Auch hier ist die Anzahl der verfügbaren Bibliotheken beträchtlich, sie vereinfachen und verkürzen das Programmieren. Deutlich stärker als beispielsweise die Arduino-IDE hat Flowcode einen Bezug zum Embedded Programming, es gilt als idealer Einstieg in Embedded C.

Atmel Studio 7 mit GCC-Compiler
Atmel Studio ist die professionelle Entwicklungsumgebung für Atmel-Mikrocontroller, und GCC ist vermutlich der meistverwendete freie C-Compiler. Auch in dieser Umgebung lässt sich die BrainBox Arduino schnell und zielgerichtet programmieren. Hier sind ebenfalls die erweiterten Arduino-Bibliotheken nutzbar, um bestimmte Funktionen zu vereinfachen.

Snap 4 Arduino
S4A macht das Ansteuern der BBA nun auch mit den beliebten Scratch Puzzleblöcken möglich. Mit der BBA stellt S4A eine stabile Verbindung her, nachdem zuvor das Programm Firmata geladen wurde. Dann werden die auf dem Bildschirm stehenden Instruktionen ähnlich einem Emulator in Echtzeit auf der Hardware ausgeführt, die mit der BBA gekoppelt ist. Die Beta-Version kann die meisten Grundprogramme in Code umsetzen, dieser kann anschließend über die Arduino-IDE geladen werden. Zahlreiche Programme sind auch offline lauffähig. Ein viel versprechender Start für S4A, ideal, um auch bei Newcomern Begeisterung zu wecken.

und eigenen 5-V-Anschlüssen auf einer Seite des Boards vorhanden (CONN2, CONN5…9). Auf der anderen Seite befinden sich die Schraubklemmen für die Motorsteuerung (CONN13), zwei digitale Ausgänge (CONN12), das I²C-Interface (CONN3) und schließlich noch eine dreipolige Gruppe (CONN11), an der die Betriebsspannungen der BBA abgegriffen werden können. Ein 6-poliger Pfostenverbinder (SERVO) ist zum Steuern von zwei Servos bestimmt, und über einen weiteren 6-poligen Pfostenverbinder (ISP) ist das Programmieren des Mikrocontrollers möglich. Auf einen 4-poligen Steckverbinder (PL3) kann ein Bluetooth-Modul HC06 aufgesteckt werden.

Auf eine Stückliste wird an dieser Stelle verzichtet, da die BBA für Lehr- und Lernzwecke sofort einsatzfähig und vorprogrammiert geliefert wird.

Zusammenfassung

Das Layout der BrainBox Arduino ist nach dem EVA-Prinzip gestaltet: Eingabe – Verarbeitung – Ausgabe. Oben befinden sich die Anschlüsse für die Sensoren, der Mikrocontroller ATMega32U4 übernimmt die Verarbeitung der Signale, und unten werden die Aktuatoren angeschlossen. Zum Anschluss von Sensoren und Aktuatoren dienen robuste Kabelschraubklemmen, sie sind als analoge oder digitale Eingänge konfigurierbar. Zu jedem Eingang gehören ein 5-V- und ein GND-Anschluss. Auch die für den Arduino spezifischen Steckverbinder sind vorhanden, dort kann ein in seinen Abmessungen standardisiertes Arduino-Shield aufgesteckt werden.

Die BrainBox Arduino verfügt über diverse Optionen für die Stromversorgung, so dass sie für die unterschiedlichsten Zwecke einsetzbar ist.

Rechts auf dem Board sind die Anschlüsse zugänglich, über die das Programmieren möglich ist. Im Lieferzustand ist die BBA mit einem Arduino Leonardo Bootloader vorprogrammiert, so dass sie über USB programmiert werden kann. Sollen Programme ohne Bootloader in den ATmega32U4 geladen werden, kann dies über den 6-poligen ISP-Steckverbinder geschehen.

Die vier Leistungsausgänge links unten auf der Platine haben PWM-Funktionalität.

Anschlüsse

In der Skizze sind die farbigen Anschlussbezeichnungen die Namen, die von der Arduino IDE und von Snap4Arduino verwendet werden. Die grauen Anschlussfelder zeigen die Bezeichnungen aus dem Datenblatt, sie sind beim AVR Studio und bei Flowcode gebräuchlich.

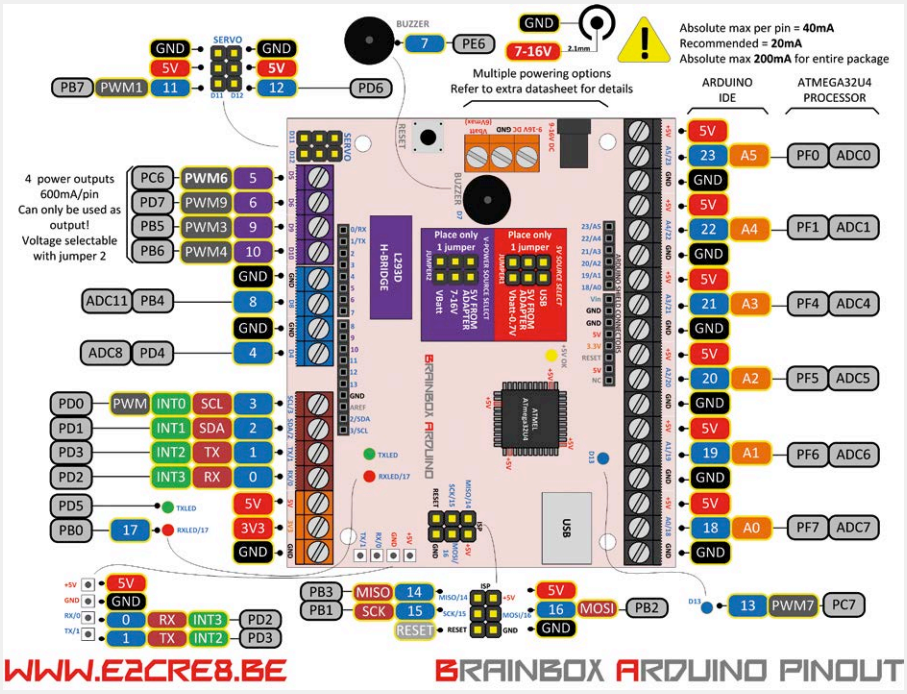


Bild 2. Anschlüsse der BrainBox Arduino mit Kurzbeschreibungen.

Sie werden über eine doppelte H-Brücke gesteuert, jede Leitung ist bis 600 mA belastbar.

Auf der rechten Unterseite sind die Anschlüsse für I²C und RS232 angeordnet, über diese Schnittstellen kann die BBA mit Sensoren, Aktuatoren oder LC-Displays kommunizieren. Der 4-polige Steckverbinder in der Ecke rechts unten macht das unmittelbare Aufstecken eines Bluetooth-Moduls HC06 möglich.

Programmbeispiele sowohl für die BrainBox Arduino als auch für APPINVENTOR geben Hilfestellung beim Schreiben individueller Apps, über die Smartphones mit der BBA kommunizieren können. Weitere Informationen, allerdings in niederländischer Sprache, sind auf der Website von E2CRE8 [1] zu finden. ◀

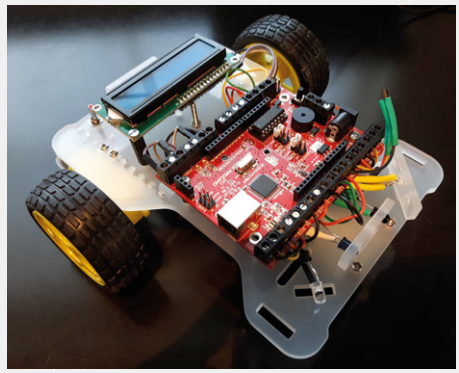
(160001)gd

Weblink

[1] http://e2cre8.be/

Bauteilesatz für BrainBox Arduino

Zum Lieferumfang des im Elektor-Shop erhältlichen Bauteilesatzes gehören die benötigten Sensoren, Aktuatoren, Leitungen, mechanische Einzelteile, Informationsmaterial und Programmbeispiele. Damit sind der Bau des abgebildeten Robotkarts und das Steuern über eine selbst geschriebene App möglich. Auch wenn es darum geht, andere attraktive und zugleich lehrreiche Projekte mit der BBA zu realisieren, steht der Kreativität kein Hindernis im Weg.



SUPRA 2.0: Die MC-Version

Signal ganz klein, Qualität ganz groß

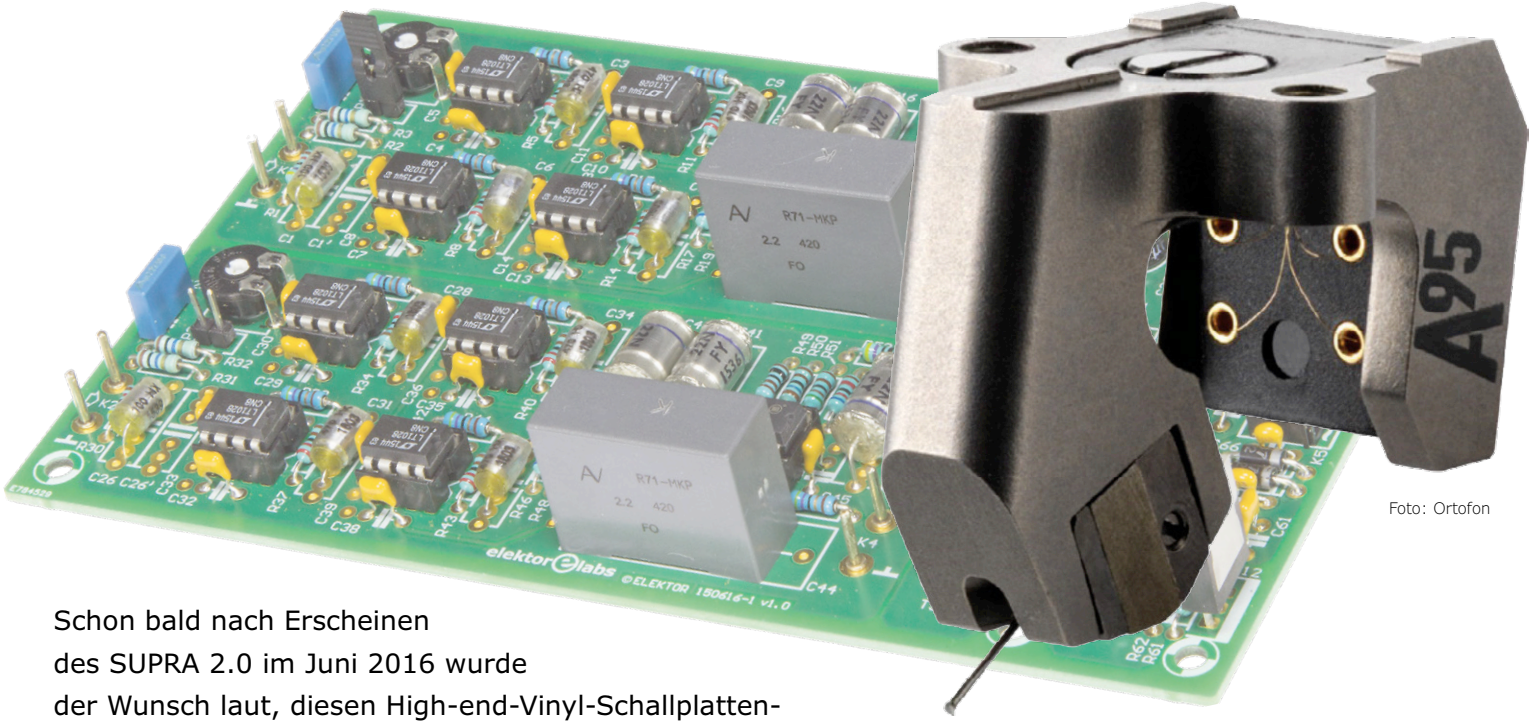


Foto: Ortofon

Schon bald nach Erscheinen des SUPRA 2.0 im Juni 2016 wurde der Wunsch laut, diesen High-end-Vinyl-Schallplatten-Vorverstärker für den Betrieb an MC-Tonabnehmer-Systemen zu modifizieren. Die Technik der parallel geschalteten Opamps bietet sich auch hier als Methode an, das Rauschen wirksam herabzudrücken. Dieser Beitrag beschreibt, was für die MC-Version angepasst oder ergänzt werden muss.

Von **Ton Giesberts** (Elektor-Labor)

Elektors MM/MD-Vorverstärker SUPRA 2.0, der im Juni 2016 erschien [1], stieß bei den Freunden der Vinyl-Schallplatte auf großes Interesse. Nicht wenige von ihnen haben sich an die Realisierung gewagt, und überraschend viele waren darunter, die im Elektor-Shop den Bauteilesatz orderten. Vermutlich lag das auch an den wertigen Komponenten zu einem moderaten Preis. In der ersten Version war der SUPRA 2.0 noch nicht für den Anschluss von Moving-Coil-Systemen (MC) eingerichtet. Was die Qualität der Wiedergabe betrifft, sind MC-Systeme den MM- und MD-Systemen grundsätzlich überlegen. MC-Systeme erfordern jedoch spezielle Vorverstärker mit höheren Verstärkungen, denn die Ausgangsspannungen betragen nur etwa 0,2...0,5 mV. Beim Entwerfen des SUPRA 2.0 in der MM/MD-Version hatten wir bereits an den Betrieb zusammen mit MC-Systemen gedacht, den Gedanken hatten wir jedoch noch nicht in die Tat umgesetzt. Inzwischen ist dies geschehen: Das Elektor-Labor hat eine MC-Version entwickelt, außerdem wurden die obligatorischen Messungen durchgeführt und dokumentiert.

Eigenschaften und Messwerte

Die Messungen wurden mit einem Eingangssignal 0,3 mV bei der Frequenz 1 kHz vorgenommen, die Quellimpedanz lag bei 5 Ω. Die Stromversorgung war ein Labornetzteil, Spannung ±14 V.

- Eingangsimpedanz: 100 Ω
- Ausgangssignal bei 0,3 mV am Eingang: 270 mV
- Signal-Rausch-Verhältnis linear: > 73 dB (B = 22 Hz...22 kHz)
  A-bewertet: > 79 dBA (B = 22 Hz...22 kHz)
- THD+N (1 kHz): < 0,02 % (nur Rauschen, B = 80 kHz)
- THD bei 1 kHz: < 0,0007 % (Mittelwert bildende FFT-Messung)
- THD bei 10 kHz: < 0,0002 % (Mittelwert bildende FFT-Messung)
- Abweichung von der RIAA-Normkurve: < 0,1 dB (50 Hz…10 kHz)
  < 1 dB (20 Hz)
  0,15 dB (20 kHz)

Modifikationen

Die vorzunehmenden Änderungen sind nicht schwierig. Betroffen sind in jedem Kanal neun Widerstände, sie müssen

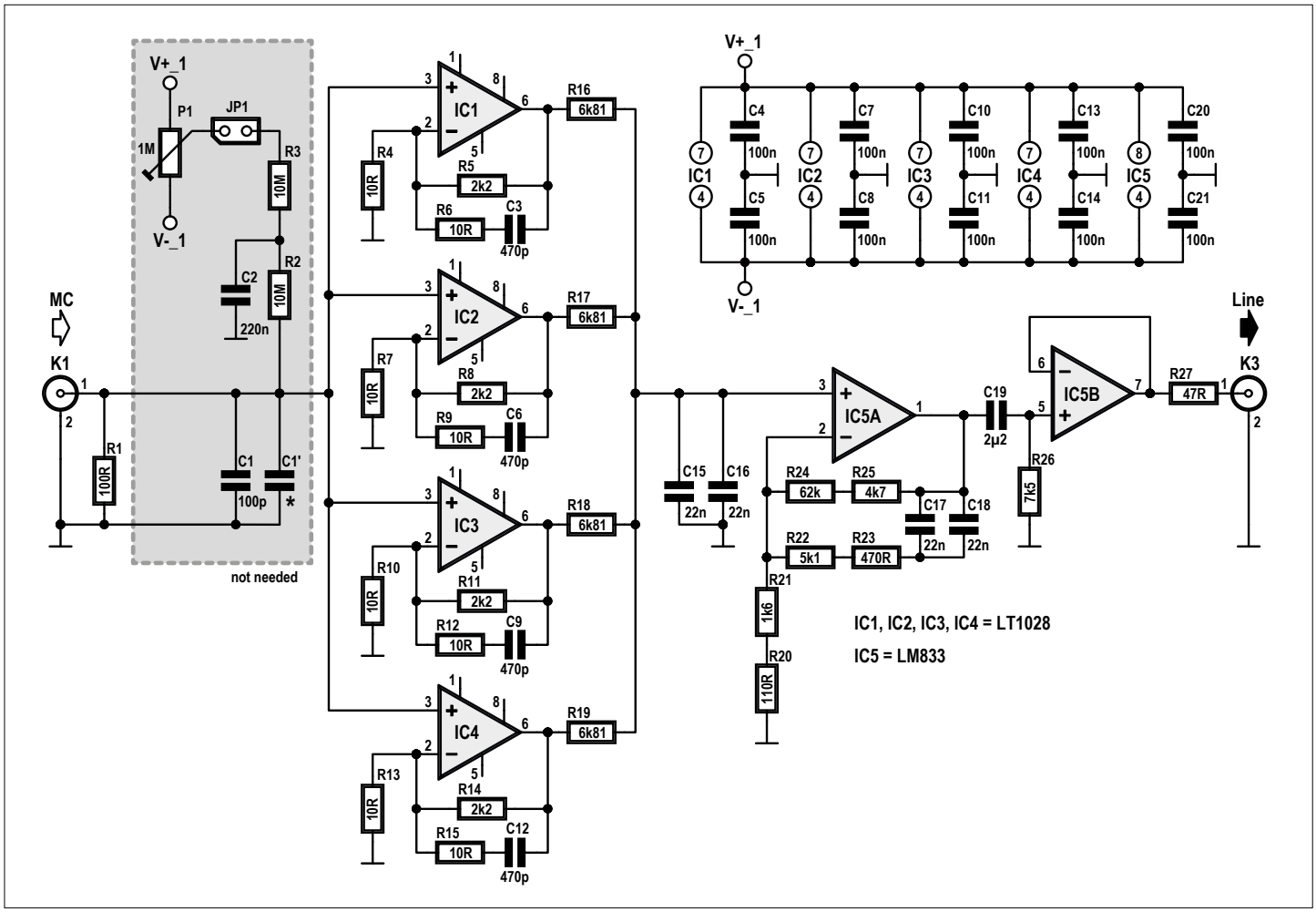


Bild 1. Schaltungsausschnitt mit geänderter Dimensionierung für den Einsatz als MC-Vorverstärker.

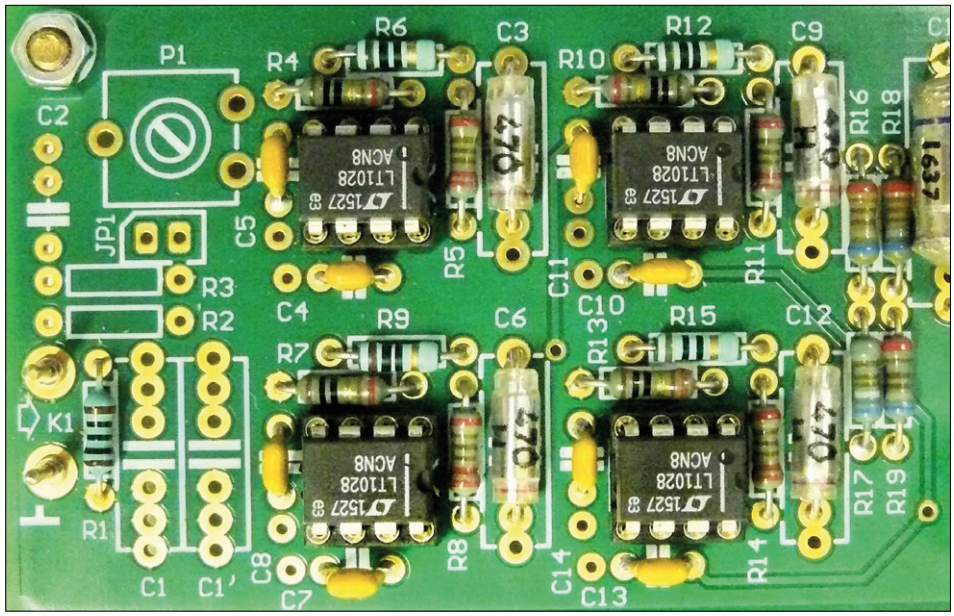
gegen Exemplare mit abweichenden Werten getauscht werden. In **Bild 1** ist der Schaltungsteil eines Kanals dargestellt, in dem sich die Widerstände befinden. Die meisten MC-Systeme sind für eine Abschlussimpedanz von nominal 100 Ω konstruiert. Deshalb erhalten die Widerstände R1 und R30, die für die Eingangsimpedanzen der Kanäle verantwortlich sind, beide den Wert 100 Ω. Falls das verwendete MC-System eine andere Abschlussimpedanz verlangt, beispielsweise 47 Ω, müssen R1 und R30 an diesen Wert angepasst werden.

Von den Kondensatoren C1 und C26 hängen die Eingangskapazitäten der Kanäle ab. Bei MM- und MD-Systemen lässt sich mit den Kapazitäten der Frequenzgang im hohen Bereich beeinflussen, MC-Systeme verhalten sich jedoch anders. Der Innenwiderstand der MC-Systeme beträgt nur wenige Ohm, die Induktivität liegt wegen der wenigen Windungen der beweglichen Wicklung im vernachlässigbaren Bereich. C1 und C26 können hier entfallen, aber falls sie schon montiert sind, dürfen sie ohne nachteilige Folgen an ihrem Platz bleiben.

Die 16 Widerstände in den Rückkoppelzweigen von IC1...IC4 und IC6…IC9, es sind R4, R6, R7, R9, R10, R12, R13, R15, R33, R35, R36, R38, R39, R41, R42 und R44, haben bisher die Werte 47 Ω. Diese Werte werden auf 10 Ω herabgesetzt, die Gesamtverstärkung des Vorverstärkers beträgt damit 900. An der Gesamtverstärkung haben die vier parallel geschalteten Opamps einen Anteil von 221.

Um die erforderliche Verstärkung zu bestimmen, wurden die Ausgangsspannungen von 26 unterschiedlichen MC-Systemen gemessen: Der durchschnittliche Wert lag knapp oberhalb 0,3 mV. Da die Line-Eingänge typischer Audioverstärker Empfindlichkeiten im Bereich 100...200 mV haben, wurde die Ausgangsspannung des Vorverstärkers auf 250 mV festgelegt. Bei diesem Wert muss die Verstärkung nicht unnötig hoch sein. Die nominale Ausgangsspannung der MM/MD-Version lag bei 468 mV, wenn 2,5 mV am Eingang lagen. Bei der MC-Version wurde die Ausgangsspannung auch noch aus einem anderen Grund herabgesetzt. Die Bandbreite der vier parallel geschalteten Opams darf auch bei hoher Verstärkung nicht unzulässig absinken. Beim verwendeten Opamp-Typ LT1028 beträgt die Bandbreite bei offener Schleife 70 MHz, mit der Verstärkung 221 liegt sie knapp über 300 kHz. Der Kondensator 470 pF in der Rückkopplung begrenzt die Bandbreite der Eingangsstufe auf 150 kHz, so dass sich eine effektive Bandbreite von insgesamt 115 kHz ergibt.

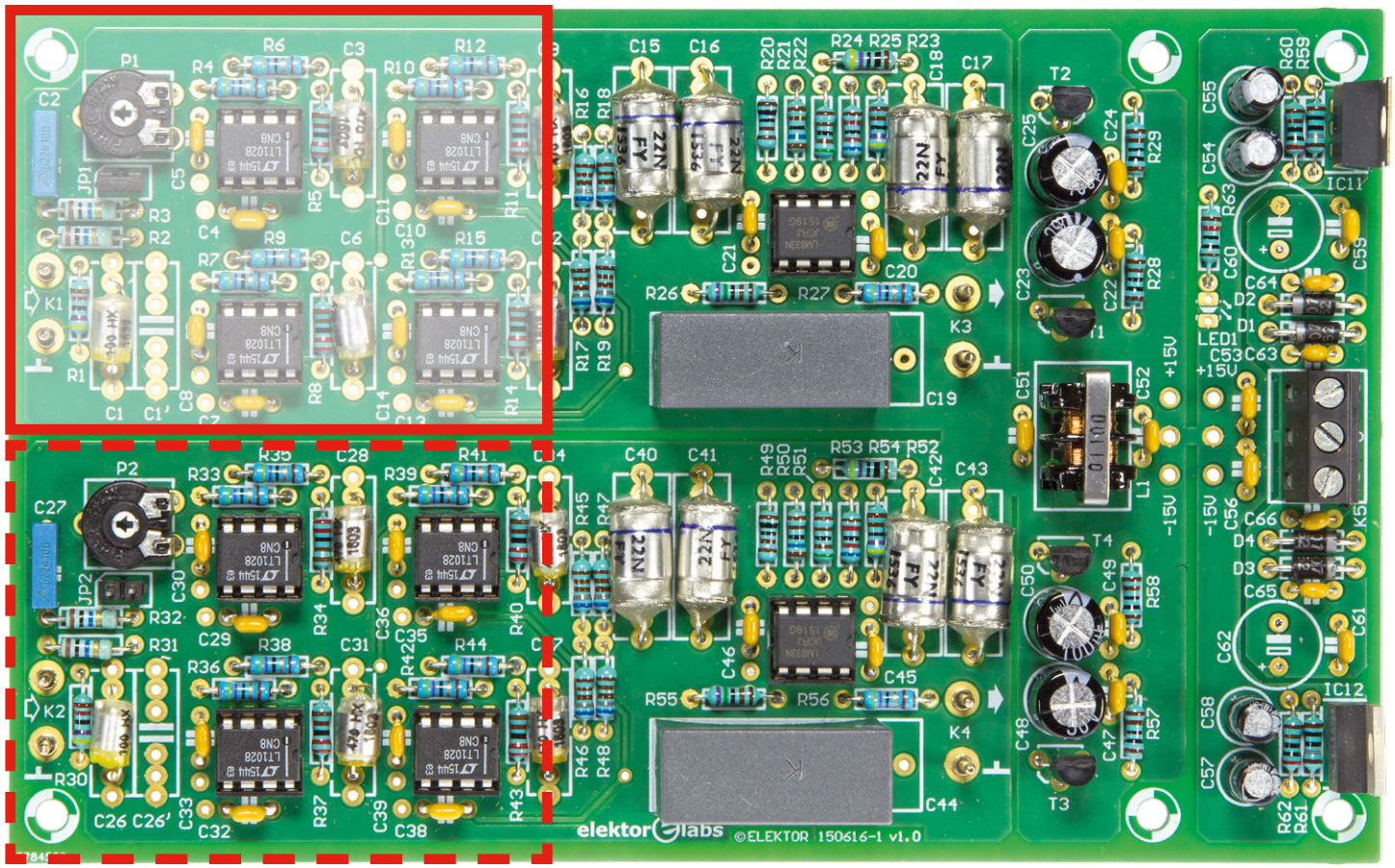
Wenn die Verstärkung der Eingangsstufen noch höhere Werte hätte, würde sich dies nachteilig auf den Frequenzgang



Anpassungen MM/MD nach MC

- R1,R30 = 100 Ω/1 % (oder abhängig vom verwendeten MC-System)
- R4,R6,R7,R9,R10,R12,R13,R15, R33,R35,R36,R38,R39,R41,R42,R44 = 10 Ω/1 %
- C1,C26 = Nicht erforderlich, können in der Schaltung bleiben, falls schon montiert.
- P1,R2,R3,C2,JP1 und P2,R31,R32,C27,JP2 = Nicht erforderlich, können in der Schaltung bleiben, falls schon montiert.

Bild 2. Platinenausschnitt mit den für den MC-Betrieb modifizierten Komponenten.



um 20 kHz auswirken. Falls im Einzelfall höhere Gesamtverstärkungen unumgänglich sind, ist es sinnvoller, die Verstärkung der zweiten, mit dem LM833 aufgebauten Stufe heraufzusetzen. Da jedoch hier die RIAA-Korrektur stattfindet, ist eine neue Dimensionierung aller Widerstände dieser Stufe nötig.

In der Schaltung des ursprünglichen SUPRA 2.0 wurden in jedem Kanal Vorkehrungen getroffen, mit denen der Eingangsruhestrom der vier Opams kompensiert werden kann. Dazu gehören die Komponenten P1, R2, R3, C2 und JP1 sowie P2, R31, R32, C27 und JP2. Diese Vorkehrungen sind bei der MC-Version wegen der niedrigen Eingangsimpedanz entbehrlich, die genannten Komponenten können entfallen. Falls sie auf der Platine bereits montiert sind, müssen sie nicht entfernt werden.

Für die MC-Version des SUPRA 2.0 empfehlen wir den Einsatz der A-Version des

LT1028. Der LT1028ACN8 liegt zwar noch etwas höher im Preis als der LT1028CN8, er bietet dafür einen deutlich niedrigeren Input-Offset- und Ruhestrom. Der höchste Wert der Offsetspannung am Ausgang, den wir mit Exemplaren des Typs LT1028ACN8 bei der hohen Verstärkung 221 gemessen haben, lag bei nur 7 mV! Die Summe der vier Offsetspannungen wird von der zweiten Stufe um den Faktor 40 verstärkt. Wenn die Offsetspannung am Ausgang von IC5A/IC10A mit Exemplaren des Typs LT1028CN8 höher als wenige 100 mV liegt, ist es eine Überlegung wert, diese Opamps gegen den Typ LT1028ACN8 auszutauschen. Die hohen Investitionen für insgesamt acht Stück können sich jedoch lohnen, denn höhere Offsetspannungen am Ausgang von IC5A/IC10A haben mehr Stromrauschen in der Gegenkopplung zur Folge. Beim Prototypen unseres SUPRA 2.0 lag die Offsetspannung am Ausgang von IC5A/IC10A bei annähernd 100 mV.

Wir wünschen Ihnen viel Freude an Ihrer Schallplattensammlung, Ihrem MC-Plattenspieler und dem High-end-Vorverstärker SUPRA 2.0! ⏮

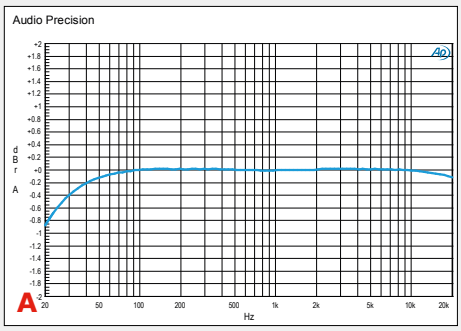
(160263)gd

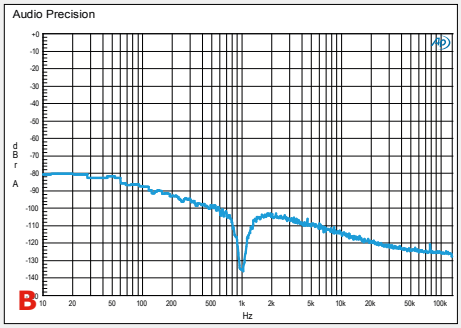
Weblinks

[1] www.elektormagazine.de/150616
www.elektormagazine.com/labs/supra-20-high-end-preamp-for-record-player-150616-i

[2] www.elektormagazine.de/labs/mc-modification-for-supra-20

Die Diagramme sollen die Messwerte ergänzen und veranschaulichen:





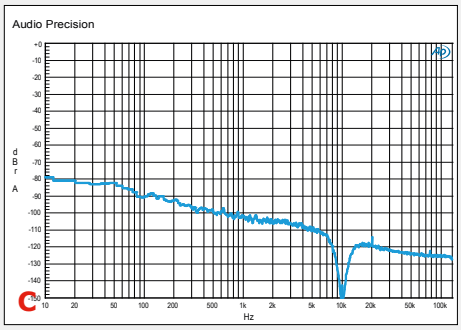
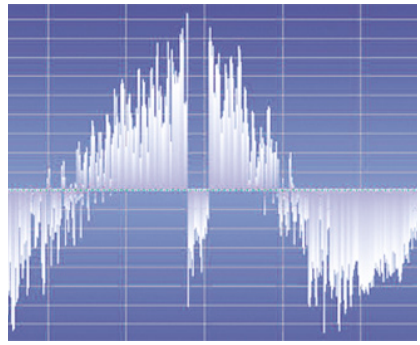


Diagramm A stellt den Amplitudenverlauf am Ausgang dar, wenn ein nach RIAA-Standard korrigiertes sinusförmiges Signal am Eingang liegt. Bei 20 Hz beträgt die Abweichung weniger als -0,9 dB, bei 20 kHz sinkt sie auf -0,1 dB. Ein geringfügiges Ansteigen der Amplitude ist um die Frequenzen 138 Hz und 3,3 kHz erkennbar. Es darf jedoch vernachlässigt werden, da die Abweichung nur +0,022 dB beträgt. Ein hervorragendes Ergebnis, zumal die für die RIAA-Korrektur verantwortlichen Komponenten Toleranzen von 1 % haben!

Diagramm B zeigt einen FFT-Plot bei einem Eingangssignal mit der Frequenz 1 kHz. Das dargestellte Spektrum läuft von 10 Hz bis 130 kHz, die Frequenz 1 kHz wird unterdrückt. Damit eine aussagefähige Kurve entsteht, wurde der Mittelwert aus 16 Messungen gebildet. Harmonische sind hier nicht sichtbar: Daraus kann geschlossen werden, dass die harmonischen Verzerrungen unter 0,0007 % liegen.

Diagramm C zeigt einen FFT-Plot wie Diagramm B, hier jedoch bei einem Eingangssignal mit der Frequenz 10 kHz. Die THD-Verzerrungen liegen ebenfalls äußerst niedrig. Sichtbar ist nur die zweite Harmonische auf dem Niveau -114 dB, was 0,0002 % THD entspricht.

Das Rauschen



Das thermische Rauschen der vier Addierwiderstände 6,81 kΩ an den Ausgängen der Opamps LT1028 hat praktisch keinen Einfluss auf das Signal-Rausch-Verhältnis der vier Opamps. Das Rauschen in einem Rückkoppelzweig (2,2 kΩ parallel zu 10 Ω) beträgt 0,41 nV/√Hz. Der Eingang eines LT1028 trägt 1 nV/√Hz bei. Beim Anschluss eines MC-Systems mit dem Innenwiderstand 5 Ω werden daraus 1,04 nV/√Hz. Daraus folgt, dass das gesamte Eingangsrauschen eines Opamps 1,12 nV/√Hz beträgt (1,08 nV/√Hz mit kurzgeschlossenem Eingang).

Am Opamp-Ausgang ist das Rauschen um den Faktor 221 höher, es beträgt 247 nV/√Hz. Wird dazu das Rauschen eines 6,81-kΩ-Widerstands addiert, steigt das Ausgangsrauschen auf 247,2 nV/√Hz. Das Gesamtrauschen hinter dem Addierverknüpfungspunkt beträgt die Hälfte dieses Werts, also 123,6 nV/√Hz. Das Stromrauschen wurde hierbei außer Betracht gelassen.

Das Rauschen der Eingangsstufe in der MC-Version liegt um den Faktor 3,7 höher als das Rauschen der (kurzgeschlossenen) MD-Eingangsstufe. Die Verstärkung ist jedoch bei der MC-Version um den Faktor 4,62 höher. Wenn das gemessene Rauschen der MD-Eingangsstufe bei kurzgeschlossenem Eingang (88 dB) auf die Ausgangsspannung 270 mV bezogen wird, sinkt das Signal-Rausch-Verhältnis um 4,84 dB auf 83,15 dB. Die rechnerische Differenz des Rauschens von MC und MD bei kurzgeschlossenen Eingängen beträgt 11,4 dB.

Aus vorstehenden Überlegungen lässt sich herleiten, dass das Signal-Rausch-Verhältnis der MC-Version des SUPRA 2.0 nahe 72 dB liegen muss. Die Messungen haben diesen Wert bestätigt.

HangTux

Hangman auf dem Raspberry Pi

Linux ist ein spannendes Betätigungsfeld. Mit Erscheinen der Einplatinencomputer wie dem Raspberry Pi sowie Beagle Bone, DragonBoard oder Odroid erhielt die Zahl der Anwender einen kräftigen Schub, denn auf diesen Plattformen läuft als Betriebssystem meistens Linux. Falls Sie auf den Trend noch nicht aufgesprungen sind, ist hier eine kleine erste Übung: Das Spiel „Hangman“.

Von **Roy Aarts** (Labor)
und **Thijs Beckers** (Redaktion)

Als kleine Fingerübung hatten wir vor einigen Jahren für das bekannte Elektor-Linux-Board das Spiel „Hangman“ programmiert. Das Spiel lief auf dem Board, und ein Terminalprogramm auf einem Linux-PC diente als Spielfläche. Der Spielstand wurde vom Elektor-Linux-Board auch auf einer LED-Anzeige sichtbar gemacht. Hier setzen wir dieses Konzept fort, an die Stelle des Elektor-Linux-Boards und des PCs ist jedoch ein Raspberry Pi getreten. Auch diesmal spielen wir „Hangman“ unter Linux, nun übernimmt der Raspberry Pi sämtliche Funktionen. Dazu muss das Raspberry-Pi-System als Ersatz-PC mit Bildschirm, Tastatur und Maus konfiguriert sein, auf dem Raspberry Pi muss eine kompatible Linux-Version laufen.

Hardware

Bild 1 zeigt die Schaltung. Ein 8-bit-Mikrocontroller MCP23S08, der über den SPI-Bus mit dem Raspberry Pi kommuniziert, steuert eine LED-Zeile. Die Platine, die wir seinerzeit entworfen haben, ist schnell aufgebaut, denn sie wird nur mit bedrahteten Komponenten und dem Mikrocontroller im DIP-18-Gehäuse bestückt. Auch Anfänger in der Kunst des Lötens werden damit zurechtkommen. Die Betriebsspannung der Schaltung wird der 3,3-V-Leitung des Raspberry Pi entnommen. Normalerweise ist dies problemlos möglich. Nur wenn das Netzteil des Raspberry Pi schon vorher an seine Grenzen gestoßen ist, muss wegen des höheren Strombedarfs auf einen leistungsstärkeren Typ zurückgegriffen werden. Die Kommunikation des MCP23S08 mit dem Raspberry Pi über den SPI-Bus findet in 24 bit breiten Worten statt. Das erste Byte adressiert den Chip und legt den Modus Lesen oder Schreiben fest. Da nur Schreibvorgänge durchgeführt werden, lautet das erste Byte 0100 0000. Und weil die Adressbits A0 und A1 stets 0 sind, können die Pins 4 und 5 von IC1 fest mit Masse verbunden werden. Das zweite Byte enthält die Schreibadresse des Registers. Hier sind nur zwei Register relevant: IODIR (I/O Direction), mit dem die I/O-Leitungen konfiguriert werden, und OLAT (Output Latch), über das ein gewünschtes Bitmuster auf die Ausgänge gelegt wird. Den Eingang /CS (Chip Select) steuert der Raspberry Pi.

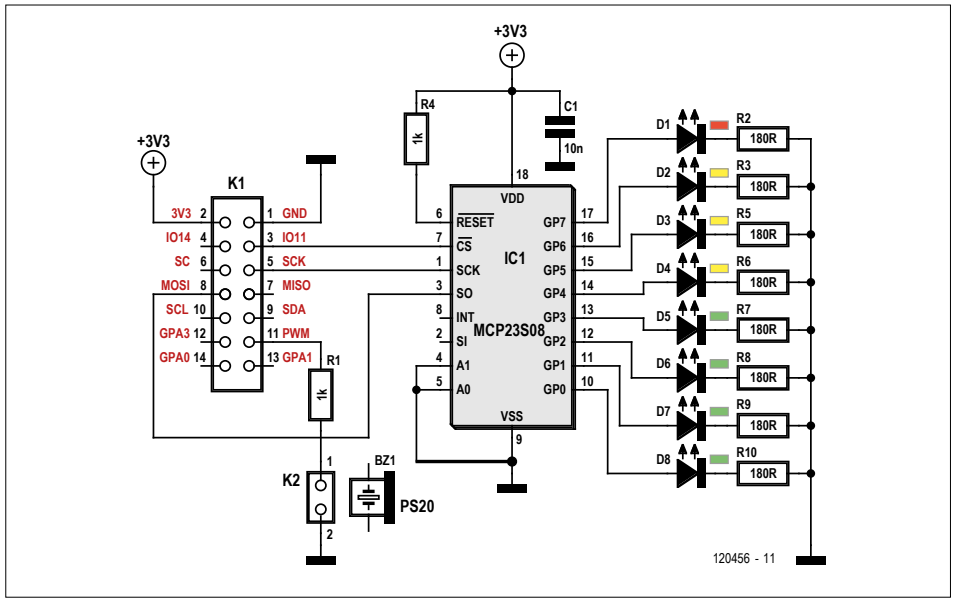


Bild 1. Die Schaltung: Ein Mikrocontroller MCP23S08 steuert die LED-Zeile, die den Spielstand anzeigt.

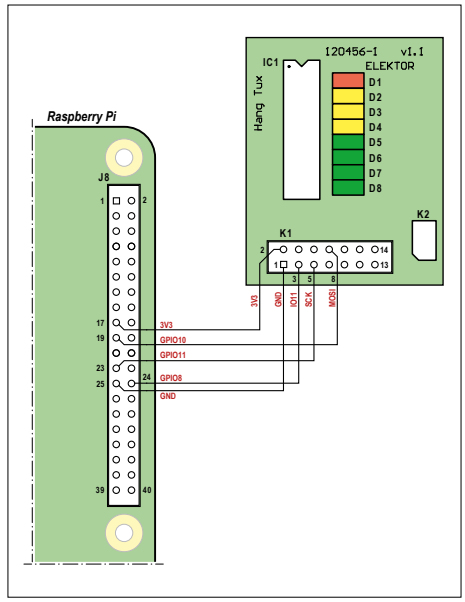


Bild 2. Verbindungen zwischen Hardware und Raspberry Pi.

Bild 3. Das Spiel startet im Hauptmenü.

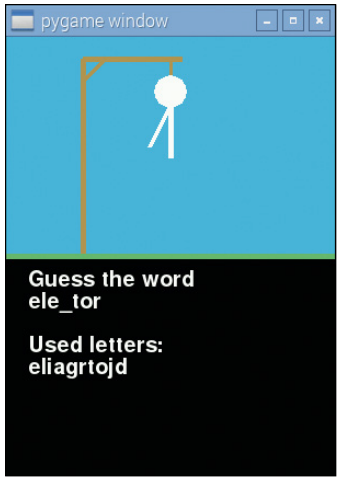


Bild 4. Wie lautet das gesuchte Wort?

Bild 5. Glückwunsch, Sie haben gewonnen!

Bild 6. Schade, spielen Sie noch einmal!

Software

Das in Python geschriebene Programm kann frei von der Projektseite [1] heruntergeladen werden. Die Installation nehmen Sie wie folgt vor. Um den SPI-Bus zu konfigurieren, öffnen Sie auf dem Raspberry Pi ein Terminalfenster, dort geben Sie ein:

```
sudo raspi-config
```

Gehen Sie dann nach *advanced -> SPI*. Die Frage, ob Sie SPI aktivieren wollen, beantworten Sie mit *ja*. Kehren Sie zum Kommando-Prompt zurück und geben Sie folgende Instruktionen ein:

```
sudo apt-get update
sudo apt-get install python-dev
   python3-dev
git clone git://github.com/doceme/
   py-spidev
cd py-spidev
sudo python setup.py install
sudo reboot
```

Übertragen Sie nun den Ordner *hangman* aus dem Download [1] zum Raspberry Pi. Das ist beispielsweise mit einem USB-Stick oder dem Programm *winSCP* möglich. Kopieren Sie den Ordner nach */home/pi*. Starten Sie dann das System neu (Reboot).
Wenn der Raspberry Pi wieder läuft, kopieren Sie die Datei *hangman.sh* aus dem Ordner *hangman* des Downloads auf den Desktop. Öffnen Sie ein Terminalfenster und geben Sie folgende Instruktionen ein:

```
sudo cd Desktop
sudo chmod +x hangman.sh
```

Verbinden Sie die LED-Anzeige wie in **Bild 2** gezeigt mit dem Raspberry Pi. Damit sind Sie startklar. Starten Sie das Spiel mit einem Doppelklick auf die Datei *hangman.sh*, die Sie auf den Desktop kopiert haben.

Eigene Wörter

Unser Spiel Hangman arbeitet mit Wörterbuchdateien. Zum Download [1] gehören bereits ein englisches und ein französisches Wörterbuch, Sie können jedoch auch Ihr individuelles Wörterbuch anlegen: Erstellen Sie eine Textdatei (.txt) mit den Wörtern, die Sie in die Wörterliste aufnehmen wollen. Jedes Wort muss in einer eigenen Zeile stehen. Geben Sie der Datei den Namen *dictionary.txt* und speichern Sie die Datei in dem Ordner, in dem das Spiel steht. Anders als bei der ersten Version des Hangman, wo das Repertoire auf 100 Wörter begrenzt war, gibt es diesmal keine Begrenzung. Das Spiel wertet als Wörterbuch immer die Datei *dictionary.txt* aus. Ein Wechsel des Wörterbuchs ist möglich, indem eine andere Wörterliste diesen Namen erhält.

Spiel ab!

Das Spiel startet im Hauptmenü (**Bild 3**) mit einem Druck auf *Enter*, hier wird es später auch beendet. Dem Muster des Galgenmanns folgend wird auf dem Bildschirm dargestellt, wie viele Buchstaben der Spieler erprobt hat, welche Buchstaben zutreffen und wie viele Versuche übrig bleiben, bis der Galgenmann vervollständigt ist (**Bild 4**). Die LED-Zeile zeigt ebenfalls die Anzahl der noch möglichen Versuche an. Für das Spiel ist die LED-Zeile zwar nicht zwingend notwendig, doch sie kann eine erste, einfache Übung sein, Hardware über einen SPI-Bus zu steuern. Und der Trend zur Elektronik mit dem Betriebssystem Linux wird sicherlich anhalten! ◀

(160127)gd

Weblinks

[1] www.elektormagazine.de/160127

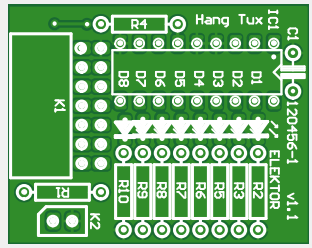
Stückliste

Widerstände:
R1,R4 = 1 k
R2,R3,R5...R10 = 180 Ω

Kondensator:
C1 = 10 n

Halbleiter:
D1 = LED rot, 5 mm rechteckig
D2...D4 = LED gelb, 5 mm rechteckig
D5...D8 = LED grün, 5 mm rechteckig
IC1 = MCP23S08

Außerdem:
K1 = Stiftkontaktleiste 2·7-polig, RM 2,54 mm
K2 = Stiftkontaktleiste 2-polig, RM 2,54 mm
Platine 120456-1



Willkommen in Ihrem E-SHOP

ELEKTOR EMPFIEHLT

Arduino-Sanduhr

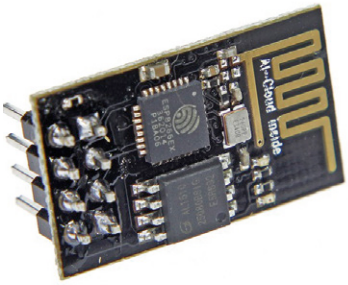
Auf der Basis eines Arduino Uno dient dieses coole Gadget zur Anzeige der aktuellen Uhrzeit. Allerdings nicht wie üblich mit LCD, LEDs oder durch die Position von zwei Zeigern, sondern indem die vier Ziffern für Stunde und Minute in eine Sandschicht „graviert" werden. Nach einer einstellbaren Zeit „löschen" zwei Vibrationsmotoren die Anzeige, bevor die neue Zeit geschrieben wird. Diese Uhr basiert auf einem Projekt des mit uns befreundeten *Make*-Magazins. Drei Miniatur-Servos steuern eine Art Pantograf. Die mechanischen Bauteile dieser Sanduhr bestehen hauptsächlich aus Streifen einer Acryl-Platte, das nicht umsonst gewählt wurde, da es preiswert und gut verfügbar ist sowie sich sehr gut per Laser ausschneiden lässt. Die Servomotoren mit metallischem Getriebe sorgen für hohe Genauigkeit mit akzeptablem Spiel.

Denis Meyer (Elektor-Labor)

www.elektor.de/sanduhr

Elektor-Bestseller

1. ESP8266-WLAN-Funkmodul
   www.elektor.de/150445-91



2. Das ESP8266-Praxisbuch
   www.elektor.de/esp8266-buch
3. Anaren Bluetooth Smart Development Kit
   www.elektor.de/anaren-bluetooth-kit
4. Elektor Uno R4
   www.elektor.de/elektor-uno-r4
5. BBC micro:bit
   www.elektor.de/bbc-micro-bit
6. AVR-Programmierung für Quereinsteiger
   www.elektor.de/avr-prog-fuer-quereinsteiger
7. SmartScope Maker Kit
   www.elektor.de/smartscope-maker-kit

Digitale Systeme mit FPGAs entwickeln

FPGAs sind Standard-ICs, die dem Anwender die Konfiguration von Hardwarestrukturen ermöglichen. Mit FPGAs ist die Umsetzung digitaler Systeme ab Stückzahl eins machbar. Durch die Verfügbarkeit kostenloser Lizenzen und preiswerter Entwicklungsboards ist die finanzielle Einstiegsschwelle in diese Technik niedrig. Bei der Überwindung der fachlichen Schwelle hilft dieser 6-teilige Kurs.

Mitgliederpreis: 49,00 €

www.elektor.de/digitale-systeme-fpga

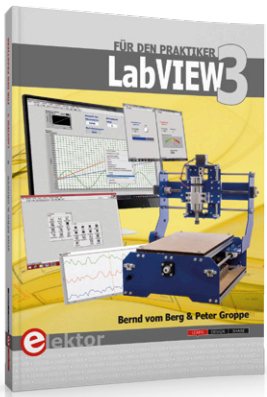
Das ESP8266-Praxisbuch

Der ESP8266 ist ein programmierbares WLAN-Funkmodul mit zahlreichen Schnittstellen wie UART, I²C und SPI. Das Board ist sehr preiswert und bereits für unter 3 Euro verfügbar. Die UART-Schnittstelle sorgt dabei für eine einfache Integration in Mikrocontrollerprojekte. Das ESP8266-Modul kann hervorragend mit dem Arduino zusammenarbeiten und ermöglicht ihm über die serielle Schnittstelle den Zugang zum Netzwerk und Internet.

Mitgliederpreis: 39,80 €

www.elektor.de/esp8266-buch

LabVIEW 3 für der Praktiker



Nachdem in den Bänden *LabVIEW 1* und *LabVIEW 2* zur Einführung in LabVIEW ein breit gefächertes und solides Grundlagenwissen vermittelt wurde, stehen mit *LabVIEW 3* fortgeschrittene Programmiertechniken auf dem Programm. Die Entwicklung eigener Sub-VIs (Unterprogramme), das Arbeiten mit Schieberegistern und Zustandsautomaten und die Ereignisgesteuerte Programmierung stellen die Schwerpunkte des dritten Bandes dar.

Mitgliederpreis: 42,00 €

www.elektor.de/labview-3

NEU

Projekte mit dem RFID-Starterkit für Arduino

Der Arduino ist inzwischen zu einer festen Größe in der Maker-Welt geworden. Der Einstieg in die Controller-Technik ist damit nicht mehr nur Experten vorbehalten. Anders sieht es aus, wenn es um Hardware-Erweiterungen geht. Hier ist der Anwender immer noch weitgehend auf sich selbst gestellt. Wenn man wirklich innovative Projekte umsetzen möchte, muss man sich direkt mit elektronischen Bauelementen befassen. Dies stellt aber viele Einsteiger vor größere Probleme.

Genau hier setzt das vorliegende Buch an, in dem es nicht nur um RFID geht. Es bietet eine Fülle an Praxisprojekten, die mit einem einzigen Kit aufgebaut werden können. Dieses (bei Elektor erhältliche) Kit, das **RFID-Starterkit für Arduino Uno**, enthält über 30 Komponenten, Bauelemente und Module aus allen Bereichen der modernen Elektronik.

Neben den einfachen Elementen wie LEDs und Widerständen sind auch komplexe und hochmoderne Module enthalten.

Mitgliederpreis: 24,90 € (frei Haus)
www.elektor.de/projekte-rfid-kit-arduino

Anaren Bluetooth Smart Development Kit

Das Bluetooth Smart Development Kit von Anaren und das innovative Online-Development-Tool Atmosphere dieser Firma zielen auf die Erstellung von embedded Code und Anwendungen für mobile Geräte, die sich per Funk mit einem embedded Bluetooth Smart Modul von Anaren verbinden. Es unterstützt das WICED Modul A20737A AIR, das auf dem Chip BCM20737 von Broadcom basiert.

Mitgliederpreis: 29,95 €

www.elektor.de/anaren-bluetooth-kit

AVR-Programmierung für Quereinsteiger

NEU

Atmel AVR ist eine 8-bit-Mikrocontroller-Familie des Herstellers Atmel. Diese Controller sind wegen ihres einfachen Aufbaus, ihrer leichten Programmierbarkeit, den kostenlosen Entwicklungswerkzeugen und der Verfügbarkeit in DIP-Gehäuseformen auch bei Makern äußerst beliebt. Darüber hinaus sind diese Controller bereits ab zwei Euro erhältlich. Im Arduino Uno-Board wird der ATmega328 verwendet.

Mitgliederpreis: 34,80 €

www.elektor.de/avr-prog-fuer-quereinsteiger

SmartScope Maker Kit

NEU

Dieses (exklusiv nur bei Elektor erhältliche) Maker-Kit enthält eine für die FPGA-Entwicklung vorbereitete Spezialversion des SmartScope mit bereits bestückten internen Headern und weiteren zur Verfügung stehenden FPGA-Pins zum Lesen oder Schreiben. Die Header haben ein Rastermaß von 0,1", was den Anschluss vieler Erweiterungsboards erleichtert.

Mitgliederpreis: 269,96 €

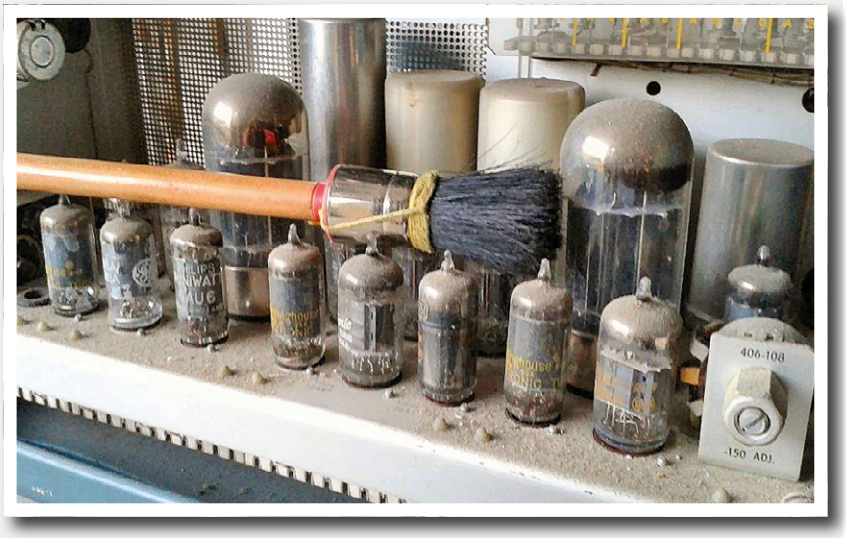
www.elektor.de/smartscope-maker-kit

Willkommen bei SHARE

Von **Thijs Beckers** (Elektor NL)

Eine denkwürdige Begebenheit *teilte* ein Entwickler unseres Labors mit mir. Er hatte zu Hause ein Problem mit seinem Heizkessel. Wenn er heißes Wasser aus der Leitung zapfen wollte, strömte einen Moment lang heißes Wasser aus dem Hahn. Dem folgte eine Weile lang kaltes Wasser, und dann wurde das Wasser wieder heiß, und es blieb heiß... Da der Heizungskessel geleast war, wurde nicht lange gegrübelt. Ein Monteur wurde bestellt, und der konnte die Ursache überraschend schnell benennen: Erst mit ziemlicher Verzögerung erkannte die Heizkesselsteuerung, dass ein Bedarf an warmem Wasser bestand. Den ersten Schub des heißen Nass lieferte ein kleinvolumiger Vorratsbehälter sofort.
So weit, so gut. Beim Kesselhersteller musste ein Ersatzteil bestellt werden, und das hieß warten, wenn nicht zwischenzeitlich der Zufall zu Hilfe gekommen wäre. Man lese und staune: Den Netzstecker um 180° gedreht und wieder eingesteckt, das Problem war behoben!
Mehr als befremdlich, denn für Wechselstrom ist die Steckerpolarität beliebig, so sollte man meinen. Der Monteur erklärte, dass das Problem *wahrscheinlich* mit der Ionisierung eines Bauteils zusammenhängt, aber Genaues weiß auch er nicht. Ich kann mir zwar vorstellen, dass die Ionisierung auf den Temperatur- oder Flusssensor tatsächlich einen Einfluss har. Aber dass die Ausrichtung des Netzsteckers eine Rolle spielt, kann ich nicht nachvollziehen. Könnte es daran liegen, dass ein Bauteil bei der Entwicklung der Kesselelektronik dem Sparzwang des Herstellers zum Opfer gefallen ist? Wenn Sie eine Erklärung haben, ich bin neugierig! Schreiben Sie bitte an redaktion@elektor.de, damit wir des Rätsels Lösung mit der Elektor-Community *teilen* können.

Nun noch etwas anderes. Als ständiger Redakteur der Rubrik Retronik kommt Kollege Jan Buiting gelegentlich mit elektronischen Röhren in Berührung. Solchen Röhren, bei denen sich die aufgedruckte Typenbezeichnung unter einer dicken Staubschicht verborgen hält. Was ist zu tun? Richtig, abstauben! Doch leider kamen in Zeiten der Röhrentechnik nicht immer die besten Glasdruckfarben zum Einsatz. Ein bisschen zu viel gerieben, und schon ist auch der Typenaufdruck wie weggeblasen!



Jan Buiting hat mir dazu ein Rezept verraten, das ich mit Ihnen *teilen* möchte. Und das lautet wie folgt: Reibe niemals mit einem Tuch über den Aufdruck, es sei denn, der Aufdruck soll verschwinden. Behandele mit einem Tuch nur die Fläche der Röhre, die nicht bedruckt ist. Dann nimm einen neuen, jungfräulichen, weichen Malpinsel und bürste **vorsichtig** und **sanft** über den noch verborgenen Aufdruck. Es kann nötig sein, 50 Mal oder mehr ohne Druck hin und her zu bürsten, bevor etwas entzifferbar ist. Zahnbürsten sind für diesen Zweck verboten, auch „weiche“ Modelle sind noch viel zu hart. Ein möglichst **weicher** Pinsel aus dem Malkasten ist hier das richtige Handwerkszeug. Anschließend säubere noch den Rest der Glasoberfläche, eventuell mit einem Tropfen Glasreiniger, wobei der Bereich des Aufdrucks konsequent auszusparen ist. Nicht noch einmal mit dem Pinsel über die feuchte Röhre und den Aufdruck fahren, anderenfalls kann alle Mühe vergeblich gewesen sein... ⏮

(160252)gd

Von Vero-Plastik zu Heavy Metal

Elektor-Laborgeräte aus den 1980ern und 1990ern

Zwischen 1984 und 1988 veröffentlichte Elektor etliche Selbstbau-Geräte für das Elektronik-Labor – alle vorgesehen für den Einbau in ein zweifarbiges Vero-Gehäuse aus Hartplastik mit Alu-Front- und Rückseite. Eine zweite Welle mit noch mehr Messgeräten folgte dann 1989 – in soliden Metallgehäusen und ebenso leicht als Teile einer Serie erkennbar. In meinem Retronik-Lager gibt es einige Exemplare aus diesen Zeiten. Nachfolgend also eine Familienchronik.

Von **Jan Buiting**, Chefredakteur Elektor EN

Irgendwann um 1995 hatte ich noch viel Wartezeit bis zu meinem Rückflug und wurde kurzerhand zu einer privaten Tour durch die RadioSpares-Laboratorien in Corby, Northamptonshire (UK), eingeladen. Auch heute noch ist das die Hauptniederlassung von RS Components. Damals gab es gerade die erste CD-Version des *RS-Katalogs*, komplett mit einer unerhört komfortablen Suchfunktion. Das war eine Sensation, auch wenn sie bei der Suche nach einem „BC547“ kläglich versagte.

Eine Aufgabe der Leute im RS-Labor in Corby war, überprüfte technische Daten von Bauteilen für die *RS Datasheet Collection* zur Verfügung zu stellen. Kunden konnten diese Sammlung abonnieren. Damals war alles auf Papier gedruckt, versteht sich. Es gab also dunkelrote Ordner, prall gefüllt mit vielen einzelnen Datenblättern. Zu meinem Erstaunen waren die fünf Labormitarbeiter mit etlichen Geräten aus der Labor-Serie von Elektor ausgestattet, die ab 1984 als Selbstbau-Projekte veröffentlicht wurden. Das war leicht an deren Vero-Gehäusen zu erkennen. Im Labor war man nach eigenem Bekunden recht zufrieden, da sie nach Kalibrierung gut funktionierten.

Es gab damals aber nicht etwa eine dezidierte „Elektor-Messgeräte-Serie“, sondern verschiedene. Der Schwerpunkt dieses Beitrags liegt auf Apparaten in zwei Gehäusen, die im Elektor-Labor zum Standard für Messgeräte gekürt wurden.

Eine Box für alle Fälle

Damals in den 1980ern, als ein Team vom Labormitarbeitern und Redakteuren bei Elektor den Plan fasste, eine Serie an bezahlbaren und doch qualitativ hochwertigen Selbstbau-Messgeräten zu entwickeln, war es keine Option, die Projekte mit dem Entwickeln der Platine abzuschließen. Wie in der Industrie üblich startete man mit dem Gehäuse und den Bedienelementen, bevor man zum Regal mit Datenbüchern und Datenblattsammlungen von RS Components schritt, um darin langwierig den geeigneten Opamp oder Steckverbinder zu suchen. Einige Entwickler beschwerten sich damals, dass das vorgegebene Gehäuse ihre Kreativität einschränken würde, aber sie machten trotzdem mit. „Eines für Alles – Alle in Einem“ war die Devise. Dies führte zu einem wiedererkennbaren Elektor-Look.

Am Anfang war Vero

Die erste mehr oder minder stringente Serie kam im zweifarbigen Vero-Gehäuse mit 205 x 137 x 75 mm (B x T x H) daher. Vielleicht erleben Sie ja ein *déjà vu*, wenn Sie den **Impulsgenerator** von **Bild 1** betrachten. Das Gehäuse besteht aus einer weißen oberen und einer grauen unteren Halbschale aus ABS. Front und Rückseite sind aus 1,5 mm starken Aluplatten gefertigt. Ich erinnere mich, dass Elektor damals bedruckte Selbstklebefolien für die Frontplatte (die zuerst gebohrt werden sollte) im Angebot hatte. Die Beschriftungen und Symbole dieser Folien wurden damals intensiv zwischen den vier Redaktionen (Niederländisch, Deutsch, Englisch und Französisch) diskutiert. Dies führte zu einem Kompromiss aus einfachem Englisch und leicht verständlichen Symbolen. Auch einen allgemeingültigen Namen zu finden war nicht so ganz einfach.

Dank passender Schlitzpaare an den Schalen links und rechts konnte man die Platinen gut senkrecht fixieren. Wie in **Bild 2** zu sehen, erlaubten in die Schlitze gesteckte Platinen die direkte Befestigung von Schaltern und Potis um a) die Bedien-Platine passend zur Frontplatte zu befestigen und b) die Bedienelemente fast „verdrahtungsfrei“ mit den Leiterbahnen dieser Platine zu verbinden. Also kein Drahtverhau, wie er damals in selbstgebauten Geräten üblich war. LEDs etc. konnten sogar schick durch transparente Bereiche der Frontplattenfolie leuchten.

Alles was mit Netzspannung zu tun hatte,

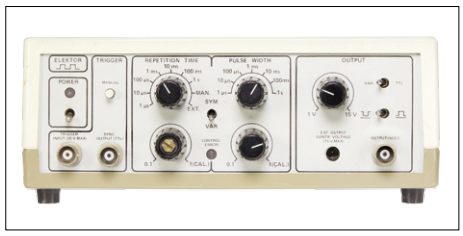


Bild 1. Messgeräte im ABS-Look: Elektor Impulsgenerator aus den 1980ern. Leider ging eine Kappe bei einem Poti-Knopf verloren.

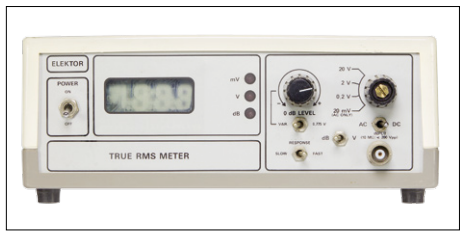


Bild 3. Nicht nur selten, sondern auch kaum benutzt: Scheinbar war der wahre Effektivwert von Spannungen nur damals in den 1980er Jahren von Interesse.

Bild 5. Im italienischen Metall-Gehäuse: Der Hochstrom-h(FE)-Tester von 1990.

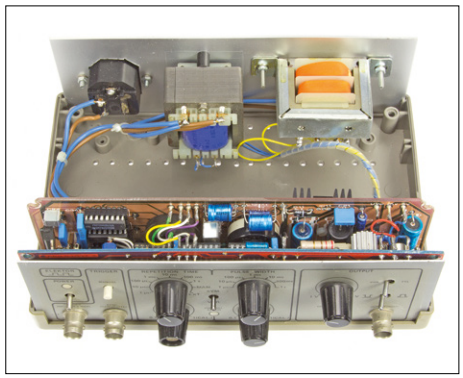


Bild 2. Das Innenleben des Impulsgenerators: Man beachte die hochkant montierten Platinen und die beiden direkt an die Rückwand geschraubten Trafos.

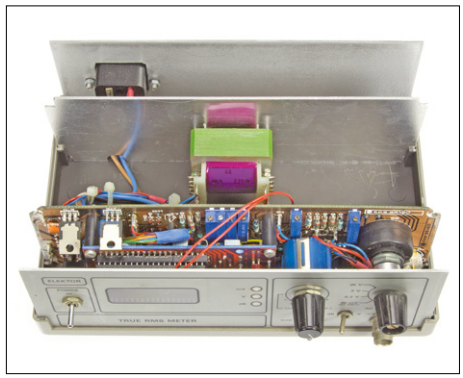


Bild 4. Innen im True RMS Meter sieht man, dass der Bereichswahlschalter direkt auf die Platine gelötet wurde.

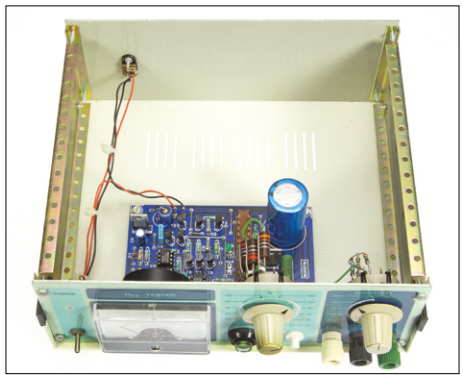


Bild 6. In diesem Telet-Gehäuse wurde ordentlich Luft verbaut!

befand sich samt Netztrafo auf der Rückseite des Gehäuses. Die Plastikbolzen für Schrauben innen an den Schalen wurden nicht genutzt, da die Platinen vertikal montiert waren. Und wo nötig, war es die Aufgabe von Jan Visser und/oder dem damals noch aktiven Herrn Feron, fein säuberlich Löcher für Kühlung etc. zu bohren.

Ein anderes schönes Exemplar aus der Vero-Serie an Messgeräten war das **True RMS Meter** von Elektor (**Bild 3**). Das LCD hat über die Jahre wohl etwas gelitten (wir sollten uns bei Sharp beschweren ;-)). Vermutlich wegen seines Gewichts wurde hier der Trafo auf eine eigene Aluplatte montiert (**Bild 4**).

Leider habe ich den prominentesten Vertreter dieser Serie nicht in meiner Sammlung: den allseits bekannten und sehr beliebten **XR2206-Funktionsgenerator**. Äußerlich sollte er sich nicht sehr unterscheiden. Man findet übrigens immer wieder noch funktionierende Exemplare dieses Klassikers auf eBay.

Ich kann mich noch an diverse andere Messgeräte dieser Serie wie etwa ein Wattmeter und ein Multimeter erinnern, aber keines von beiden ist im Retronik-Lager zu finden.

Das Vero-Gehäuse war leider nicht gerade preiswert. Elektor bekam dafür Kritik aus den eher sparsamen Ecken Europas. Einige der damals in Elektor inserierenden Händler stellten komplette eigene „Elektor-Bausätze“ oder Bauteilesätze zusammen und boten sie an. Bekannt für diesen Fleiß waren vor allem die Firmen Maplin (UK), DIL Elektronica (NL) oder Geist (D). Nichteuropäische Länder wurden von C-I Electronics bedient. All diese Firmen trugen zum Erfolg und zur großen Verbreitung dieser Geräte bei.

Italienischer Stil

Ganz wie industrielle Hersteller hat auch Elektor an Verbesserungen und an einer neuen Produktlinie gearbeitet. Ab etwa 1988 galt das Vero-Gehäuse nicht mehr als zeitgemäß und es begann die Suche nach einem neuen Standard-Gehäuse für Selbstbau-Messgeräte. Nach wie vor galten die Kriterien: günstig — gut — professionell.

Die Gerüchte besagen, dass der Distributor Texim Electronics, ein Bauteilelieferant aus einer Gegend, die in den Niederlanden als *Achterhoek* (hinterste Ecke) verspottet wird, „ganz uneigennützig“ Gehäuse der italienischen Firma Telet vorgeschlagen hat. Anders als die Vorgänger von Vero waren diese a) ganz aus Metall und b) in verschiedenen Größen verfügbar. Also wurden sie zum neuen Standard.

Bild 5 zeigt ein nettes Beispiel aus der neuen Serie von Elektor: den **Hochstrom-h(FE)-Tester** mit den Maßen 198 x 180 x 80 mm (B x T x H). Man beachte die korrekte Tiefstellung auf der Frontplatte – nachträglich Dank an Patrick Wielders aus der Zeichenabteilung! Außerdem können Sie zwei schöne Pastellfarben auf der Frontplattenfolie bewundern, wenn die Drucktechnik das halbwegs authentisch rüberbringt. Bei den neuen Messgeräten im „italian style“ war die Montage der Platinen auch horizontal möglich. Die Bedienelemente kamen direkt auf die Frontplatte und mussten per Litze mit der Platine verbunden werden (**Bild 6**). Hintenrum ist beim Hochstrom-h(FE)-Tester ziemlich viel Luft verbaut, da aus Sicherheitsgründen kein internes, sondern ein externes Netzteil zum Stecken vorgesehen war.

Welche Telet-Gehäuse finden sich noch in der Retronik-Sammlung?

- Ein **Wirkleistungsmessgerät** für

den Leistungsbezug aus dem Netz mit LCD, schaltbaren Bereichen ×1 und ×10 und einer richtigen Steckdose auf der Frontplatte (**Bild 7**) mit den Abmessungen 198 × 132 × 80 mm [2].

- Ein **80C535-Powermeter** von 1993 im größten je von Elektor verwendeten Telet-Gehäuse mit stolzen 297 × 180 x 80 mm (**Bild 8**) [3]. Besonderer Luxus für ein Selbstbaugerät: Es gab eine serielle V24-Schnittstelle, mit der man Daten auslesen und das Gerät kalibrieren konnte. Die neunpolige Sub-D-Buchse befindet sich auf der Rückseite.
- Den **TV Pattern Generator** benutze ich heute noch, um meine antiken Oszilloskope auf ihre Trigger- und Verzögerungsfähigkeiten zu testen (**Bild 9**). Die Abmessungen: 247 × 180 × 80 mm. Die LEDs stecken in Plastik-Fassungen und stehen etwas aus der Frontplatte heraus. Hier wurde ein internes Netzteil verbaut.
- Zum Schluss noch das **Inductance Meter** – das immerhin Nanohenry messen kann (**Bild 10**). Natürlich wird es gerade in den kleinen nH-Bereichen vom neueren **500-ppm-LCR-Meter** geschlagen.

Zwei weitere Messgeräte dieser Serie werden zwar nicht gezeigt, sollen aber zumindest erwähnt werden: Das einfache **Netzteil mit 15 V und 500 mA** und das **High-Tech-LCR-Meter** [4]. Letzteres habe ich als DIL-Elektronica-Bausatz von einem mittlerweile leider verstorbenen Elektor-Leser gekauft.
Ich musste das tolle Teil von Hans Bonekamp noch zusammenbauen und dann mit dem neueren 500 ppm LCR Meter vergleichen – das aber gut 2,5 dB mehr kostet.

Unvergesslich

Außer dem True RMS Meter funktionieren alle hier vorgestellten Messgeräte nach wie vor prima und sind daher Teil der *Retronics Classic Repair Initiative*, die ich bei Elektor ins Leben gerufen habe. Auch wenn mir klar ist, dass ihre kombinierte Messleistung von einem Red Pitaya in ein paar Taktzyklen abgearbeitet werden kann, und heute jeder Fünfzehnjährige sowas mit einem Arduino und einer Sammlung preiswerter Shields in einem Nachmittag oder zwei realisiert, bereiten diese Retro-Apparate immer noch viel Freude. Beispielsweise ist das Wirkleistungsmessgerät permanent in die Zuleitung zu meinen Retronik-Schätzchen eingeschleift, um mich gelegentlich an deren Energieverbrauch zu erinnern. Denn da kommen schnell mehr als 1.000 W zusammen.
Die Elektor-Artikel zu diesen Messgeräten beleuchten in der Regel umfassend die dahintersteckenden Messprinzipien und gehen in gewohnt eleganter Weise ganz nebenbei auf die konkrete Technik ein. Und dann kommt noch der Aufwand hinzu, den das Entwickler-Team in die Mechanik und die Optimierung des konkreten Einsatzes investiert hat.
Falls Sie noch ein hier nicht erwähntes Elektor-Messgerät im Vero- oder Telet-Gehäuse herumstehen haben: Bitte melden Sie sich, denn die hier präsentierte Liste ist nicht erschöpfend. ◀

(160197)

ESTD 2004

www.elektor.tv

Retronik ist eine monatliche Rubrik, die antiker Elektronik und legendären Elektor-Schaltungen ihre Referenz erweist. Beiträge, Vorschläge und Anfragen telegrafieren Sie bitte an Jan Buiting (editor@elektor.com).

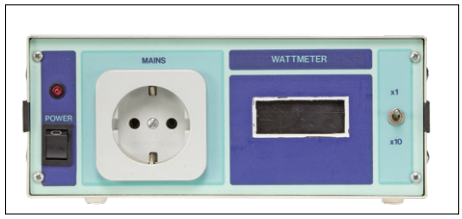


Bild 7. Das fast spartanisch aussehende Wirkleistungsmessgerät von 1991.

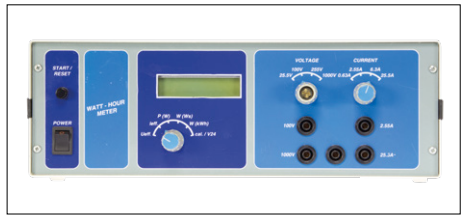


Bild 8. Das Elektor-Messgerät mit dem größten Telet-Gehäuse war das 80C535-Powermeter von 1993.

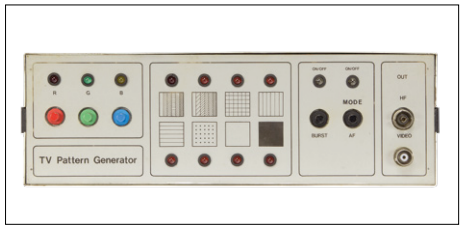


Bild 9. Immer noch regelmäßig im Retronik-Labor benutzt: Der TV Pattern Generator steuert aber heute kaum noch Fernseher an.

Bild 10. Inductance Meter: Es fällt schwer nachzuvollziehen, warum hier ein winziger Punkt statt dem Zeichen „x" für die Bereichswahl verwendet wurde.

Weblinks:

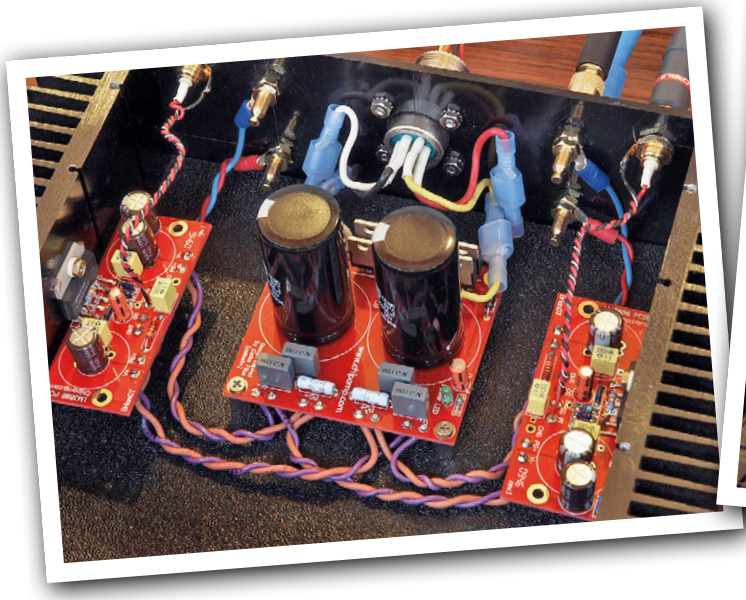
Hinweis: Die originalen Artikel aus den 1980ern und 1990ern sind auf den DVDs Elektor 1980 - 1989 und 1990 - 1999 enthalten.

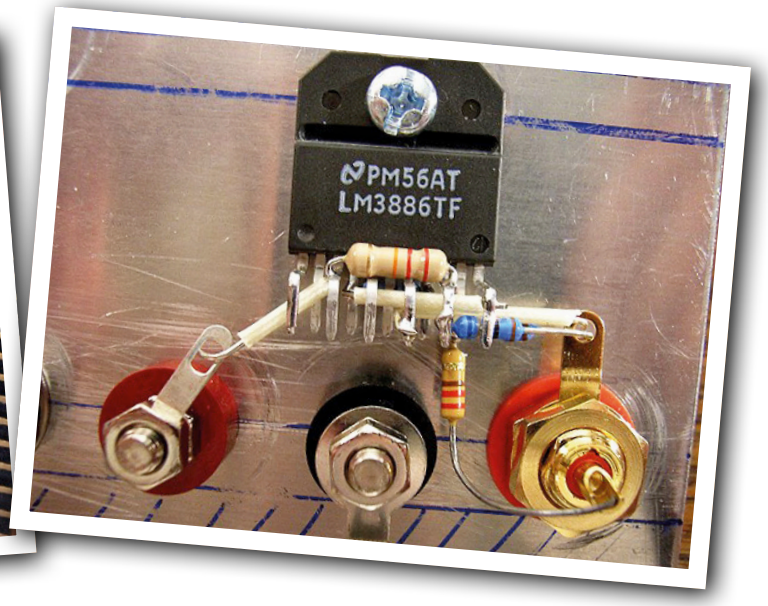
[1] **Hochstrom-h(FE)-Tester, Elektor September 1990**:
www.elektormagazine.de/magazine/elektor-199009/29390

[2] **Wirkleistungsmessgerät, Elektor März 1991**:
www.elektormagazine.de/magazine/elektor-199104/29545

[3] **80C535-Powermeter (1), Elektor Februar 1992**:
www.elektormagazine.de/magazine/elektor-199212/30063
80C535-Powermeter (2), Elektor Januar 1993:
www.elektormagazine.de/magazine/elektor-199301/30081

[4] **High-Tech-LRC-Meter (1), Elektor April 1997**:
www.elektormagazine.de/magazine/elektor-199704/31189
High-Tech-LRC-Meter (2), Elektor Mai 1997:
www.elektormagazine.de/magazine/elektor-199705/31206





Bauen Sie einen Chipamp!

Toller Sound mit wenig Aufwand

Audio-Equipment selbst bauen und finetunen – das füllt die Freizeit vieler Mitmenschen aus. Nichts ist schöner als das Erfolgserlebnis nach dem gelungenen Selbstbau und das Genießen des Sounds, der aus einer neuen Installation erklingt. In dieser Folge unserer Web-Recherchen geht es darum, dass ein anspruchsvoller Endverstärker nicht unbedingt einen komplexen Entwurf mit zahllosen Bauteilen voraussetzt.

Von **Harry Baggen** (Elektor-Labor)

Längst nicht immer sind Audioenthusiasten auch Elektroniker. Den meisten liegt das Sounderlebnis am Herzen, der technische Aspekt rückt in den Hintergrund. Den Klängen lauschen, die den Ohren schmeicheln, das ist es, worum es geht. Dieser Nimbus hat wohl dazu beigetragen, dass die Audiowelt erstaunliche Produkte hervorgebracht hat, die einem technischen Redakteur das eine oder andere Fragezeichen aufzwingen. Beispielsweise hölzerne Unterleger für die Lautsprecherkabel...

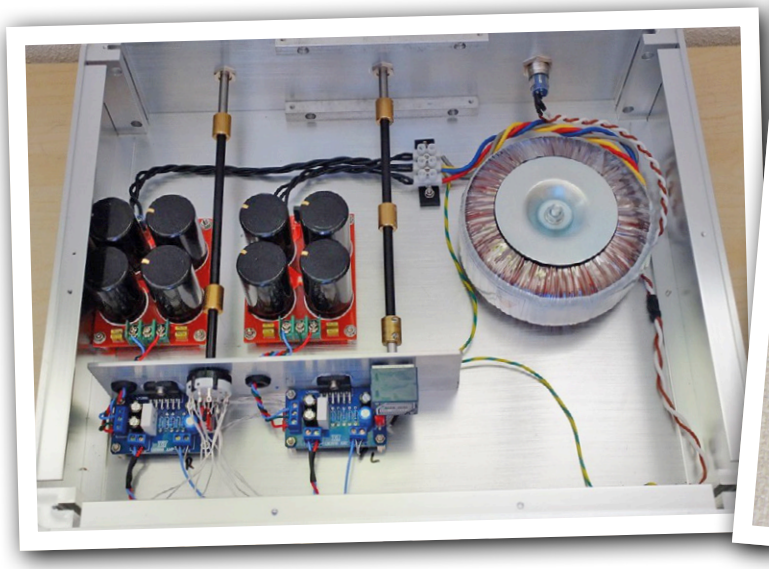
Beim Auswählen der Bauteile für einen Audioverstärker lässt sich lange darüber sinnieren, mit welcher Konstellation der Klang optimal ist. Auch das Schaltungskonzept eines Audioverstärkers ist schon lange ein heiß diskutiertes Thema. Bei unseren Spaziergängen im Internet fiel uns auf, dass sich überraschend viele Websites mit so genannten *Chipamps* oder *Gainclones* beschäftigen, einfach gestrickte Audioendverstärker, die auf integrierten Modulen basieren. Meistens sind dies Produkte aus der Familie *Overture* von National Semiconductor, inzwischen übernommen von Texas Instruments.

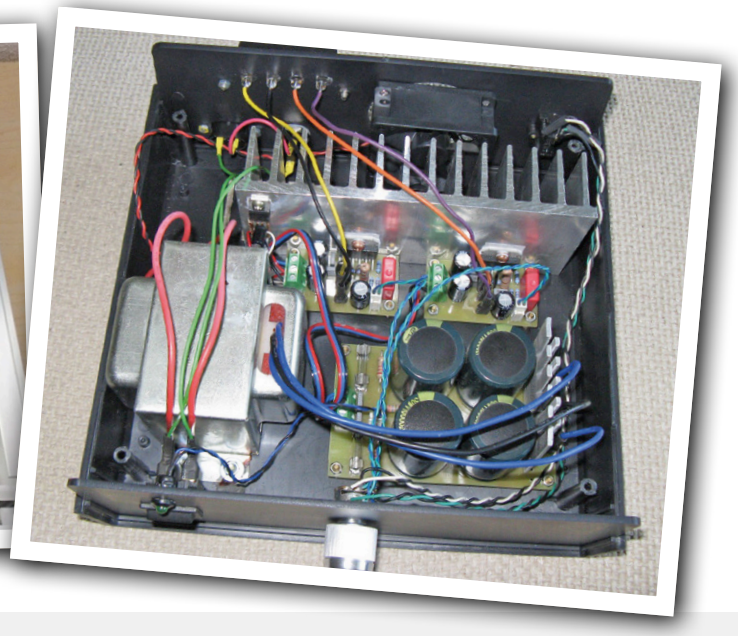
Die Verbreitung dieser Familie, die eigentlich für allgemeine Audioanwendungen und nicht für HiFi-High-End-Spitzengeräte entwickelt wurde, begann 1999 mit dem Verstärker 4706, genannt Gaincard, des japanischen Herstellers *47 Labs* [1]. Die Philosophie dieses Spezialisten folgt einem minimalistischen Leitsatz: Je weniger Komponenten im Signalweg, desto reiner der Klang. Der 4706 begnügt sich mit nur neun Bauteilen pro Kanal. Das Kernstück ist ein LM3875, ein IC, das nur wenige Euro kostet. In vielen ernst zu nehmenden Reviews wurde der mehrere tausend Dollar teure Verstärker in den Himmel gepriesen, auch heute ist das nicht anders. Damals gingen einige Konkurrenten aus dem HiFi-Segment dazu über, das IC auch in ihren Produkten einzusetzen.

Nach dem Erfolg des 4706 fühlten sich zahllose Audio-Selbstbauer inspiriert, mit der IC-Familie *Overture* zu experimentieren. Im Internet existieren sogar Webshops, die ausschließlich Bauteile und Platinen rund um *Overture* anbieten. Natürlich sind auch die Chinesen dabei, dort kann der Kunde für weniger als 10 $ eine aufgebaute Platine mit dem LM3886 erstehen [2]. Das ist das leistungsstärkste Mitglied von *Overture*, es glänzt mit fast 70 W an 4 Ω. Auch die passende Stromversorgung ist in China spottbillig zu haben. Bei den Preisen muss aber die Frage gestattet sein, von welcher Qualität die Bauteile sind.

Integrierte Audioverstärker dieser Kategorie laden zum Experimentieren mit unterschiedlichen Ausführungen und Qualitäten der Bauteile ein. Fügen Sie einen Trafo, einen Brückengleichrichter und einige Elkos hinzu – schon macht der Verstärker Musik. Wie klingt er mit anderen Elkos, nach Parallelschalten vieler kleiner Elkos, mit einem anderen Kondensator am Eingang? Dem Forscherdrang sind keine Grenzen gesetzt.

Eine aufschlussreiche Beschreibung diver-





ser Experimente mit einem Chipamp LM3886 findet man auf der Website von Dick van de Merwe [3], allerdings in niederländischer Sprache. Auf der Website von DIY Audio Projects sind ebenfalls verschiedene Projekte rund um Chipamps zu finden, wie der DIY LM3875 (Beiname *The Beast*) [4] sowie das Kit DIY LM3886 Chip Amplifier (Beiname *Gainclone*) [5]. Dass es auch ohne Platinen geht, beweist Mick Feuerbacher mit einer umfangreichen Fotostrecke auf seiner Website Dogbreath [6]. Selbstverständlich hat auch Elektor einen Beitrag mit dem LM3886 veröffentlicht, im Juli/August 1998 [7], als das IC gerade auf dem Markt erschienen war. Sogar eine Platine haben wir damals entworfen. Das Internet hält natürlich noch viel mehr Lesestoff zum Thema *Overture* bereit. Bei allen Varianten geht es um ICs, an die nur einige passive Komponenten gelötet werden müssen.

Ach, und noch etwas: Wir halten es für müßig, über die Philosophie von Audioverstärkern zu streiten... ⏮

(160261)gd

Weblinks

[1] www.sakurasystems.com/reference.html

[2] www.ti.com/product/lm3886

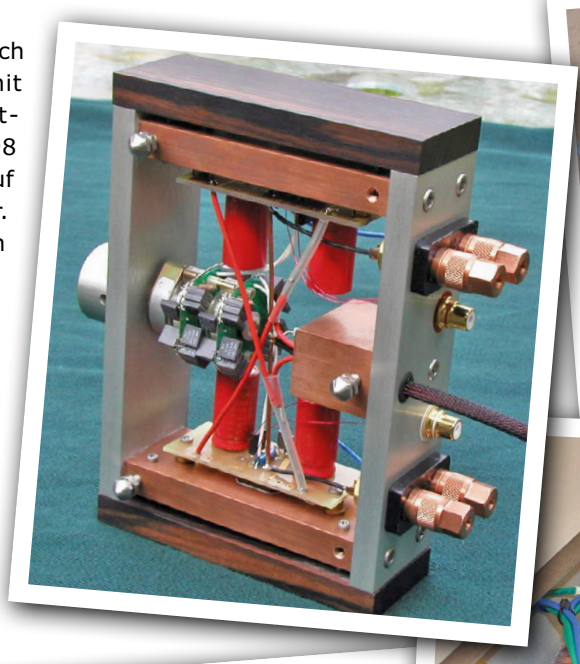
[3] www.audio-creative.nl/audio-creative/diy-lm3886-chipamp/

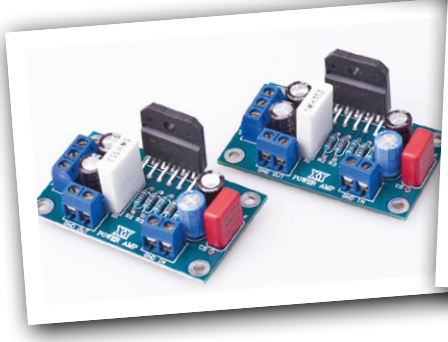
[4] http://diyaudioprojects.com/Chip/Beast/

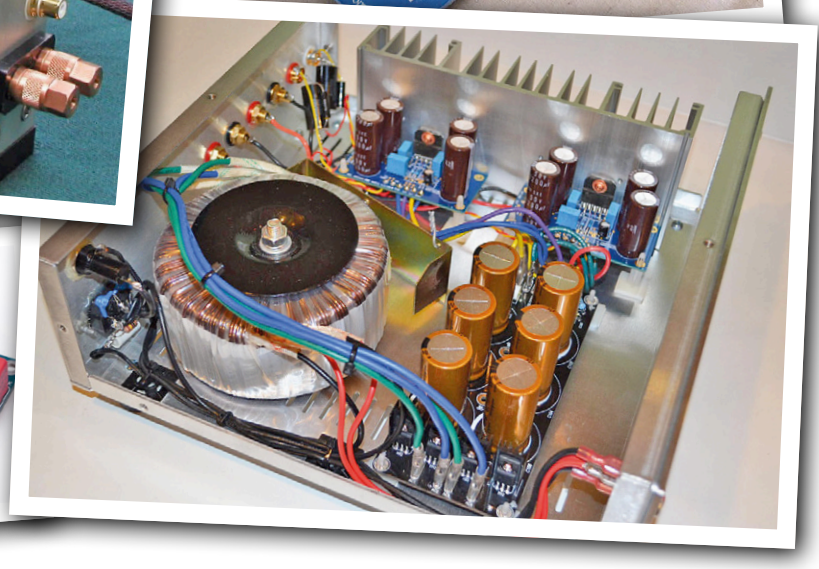
[5] http://diyaudioprojects.com/Chip/LM3886_CA/LM3886_CA.htm

[6] http://dogbreath.de/

[7] www.elektormagazine.de/magazine/elektor-199807/39291







Anschlüsse eines Opamps…

Willkür oder Logik?

Was kann nicht alles schief gehen bei der Montage eines Bauteils auf eine Platine? Alles, das ist doch klar. Doch in diesem Fall war der Grund mehr als befremdlich: Bei einem Opamp waren die Anschlüsse der Betriebsspannung vertauscht!

Von **Luc Lemmens** (Elektor-Labor)

Beim Testen des ersten Prototyps der „Tabula Tangibles“ (ein Gemeinschaftsprojekt des Elektor-Labors und der RWTH Aachen) schien mit der Stromversorgung ernsthaft etwas im Argen zu liegen: Der Ausgang des Spannungsreglers verhielt sich so, als ob er kurzgeschlossen wäre.

Zunächst lag der Verdacht auf einen Lötfehler nahe. Wenn eine Platine von Hand mit SMDs bestückt wird, schleichen sich gern Lötbrücken zwischen den IC-Kontakten ein, die nicht immer sofort auffallen. Das ist nicht weiter schlimm, mit etwas Entlötlitze sind solche Fehler schnell behoben, vorausgesetzt sie sind gefunden! Eine akribische visuelle Inspektion der Platine lieferte jedoch kein Ergebnis. Auch das Überprüfen des Platinenlayouts und der Anschlüsse der Bauelemente lief ins Leere. Verdächtig war nur ein ungewöhnlich niedriger Widerstand zwischen dem Ausgang des Spannungsreglers und Masse, da konnte irgendetwas nicht stimmen.

Eine bewährte Methode, solche Fehler aufzuspüren, ist trivial, zeitintensiv und effizient: Die Bauteile, die potentielle Verursacher sein könnten, Stück für Stück von der Platine demontieren und jedesmal den Widerstand der Betriebsspannungsleitung nach Masse messen, bis der Übeltäter entlarvt ist.

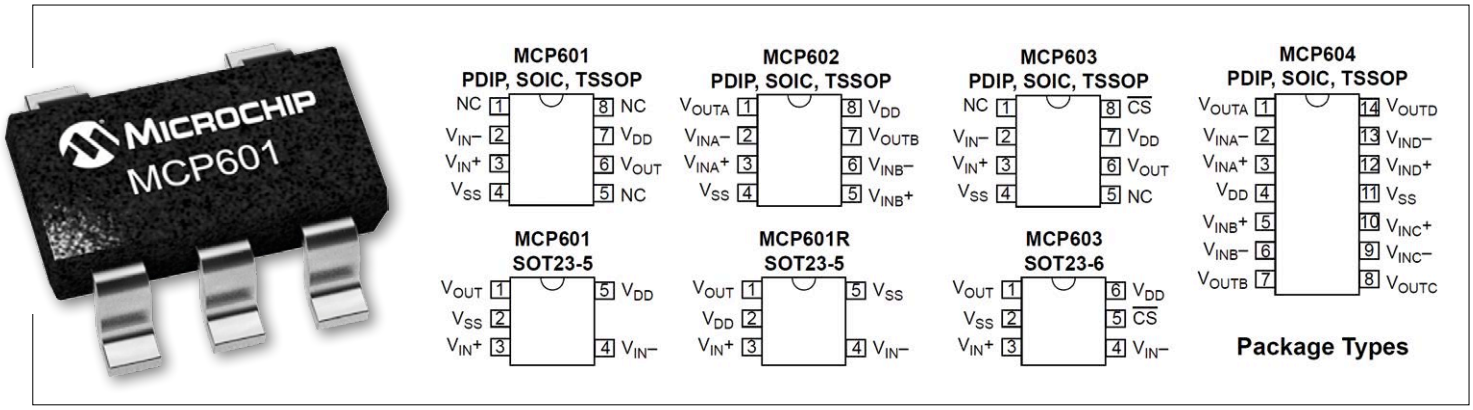
Ich hatte Glück, denn nur wenige Bauteile auf dieser Platine waren verdächtig, und ich hatte noch einmal Glück, denn schon nach dem Entlöten des ersten ICs war der Kurzschluss behoben. Es handelte sich um einen Opamp des Typs MCP601 von Microchip im Gehäuse SOT23-5. Natürlich wurde sofort das Datenblatt zu Rate gezogen, und siehe da: Es existieren zwei Varianten dieses Opamp, MCP601 und MCP601R, die Gehäuse sind identisch! Der Unterschied besteht lediglich darin, dass die Betriebsspanungsanschlüsse VDD und VSS gegeneinander vertauscht sind.

In der Elektronik habe ich schon viel Ungereimtes erlebt. Zum Beispiel die Transistoren BF254 oder BS170, bei denen ich wissen musste, von welchem Hersteller sie kamen. Nur so konnte ich sicher sein, dass die Anschlussreihenfolge stimmte. Das Suffix hinter der eigentlichen Typenbezeichnung deutet bei ICs häufig auf die Gehäusebauform hin, das wusste ich. Doch dass ein „R” ein Zeichen für vertauschte Anschlüsse VDD und VSS ist, das war mir neu! Mir fiel spontan kein plausibler Grund ein, weshalb ein Hersteller diesen Unterschied überhaupt macht.

Wie immer war die Zeit reichlich knapp bemessen, so dass ich dem Phänomen der vertauschten Anschlüsse beim MCP601 zunächst nicht weiter nachgehen konnte. Das tat ich erst, als ich der Redaktion versprochen hatte, darüber etwas zu schreiben. Trotzdem fehlt mir zu dieser Frage noch immer die Antwort: Weshalb produziert ein Hersteller zwei Varianten des gleichen IC-Typs, die sich ausschließlich durch vertauschte Anschlüsse der Betriebsspannung unterscheiden?

Wer darin eine Logik sieht, möge sie mir erklären!

(160257)gd

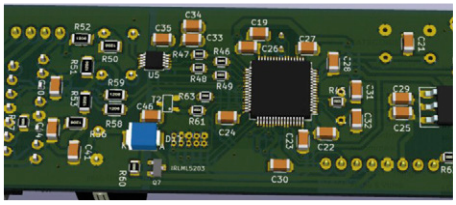


Elektor Labs Pipeline

2017 hat begonnen – Anlass für uns, hier wieder eine Liste innovativer und hoffnungsvoller Projekte zu präsentieren. Damit können wir gut ins neue Jahr starten!

Bauen Sie einen LOUD-Soundplayer!

Auf der Suche nach einem leistungsstarken Gong oder akustischen Melder? Das hier vorgestellte Projekt „Card Sound“ besteht aus einem Audio-Verstärker TDA7266 mit zwei 7-Watt-Ausgängen, einem A/D-Wandler CS4344, einem ARM-Cortex-M4-Mikrocontroller STM32F401 und einem Slot für eine Micro-SD-Karte. Damit lassen sich verschiedene, qualitativ hochwertige, auf der SD-Karte gespeicherte Sounds mit einer Lautstärke abspielen, bei der auch der tiefste Schläfer aus dem Bett fällt.



https://goo.gl/yc4or6

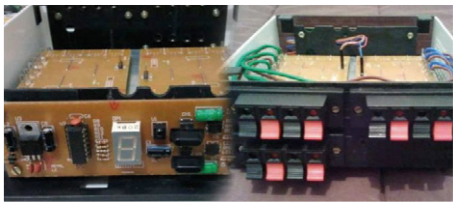
Raspberry Pi als Gärtner

Notwendig, aber lästig ist im Sommer die Bewässerung des Rasens. Deshalb sollte man sich schon im Winter darauf vorbereiten und einen automatisierten Rasensprenger entwerfen. Ein erster Prototyp eines 12-Kanal-Sprinkler-Timer/Controllers mit einem intelligenten PIC-basierten und touchfähigen Grafikdisplay funktionierte schon gut, konnte aber nicht ferngesteuert werden. Die zweite Version ist technologisch weiter fortgeschritten und besteht aus einem PIC32, Ethernet, Windows 10 IoT und einem Raspberry Pi.

https://goo.gl/JwND75

Ferngesteuerte Lautsprecherbox

Manche Leute möchten 5+1-Surround-Audio hören, aber nicht ihre Stereoanlage aufgeben. Bedauerlicherweise würden beide Systeme zusammen acht Lautsprecher erfordern, was für viele zu viel und auch nicht notwendig ist. Da beide Systeme L- und R-Lautsprecher besitzen, entstand die Idee, mit einer Schaltbox zwischen 5+1 und Stereo umschalten zu können, so dass zwei der acht Lautsprecher entbehrlich werden.



https://goo.gl/6h6oJos

Intelligenter Batterieanalysator Intelli-Cell

Hier ist ein sehr intelligentes und programmierbares Akku-Analyse- und Ladegerät mit einer speziellen Halterung, die mechanisch, chemisch (NiCd, NiMH, Li-Ion, Li-Fe, Bleigel) und elektrisch (von 1,25 V bis 12 V) für jede Handybatterie (einschließlich Apple) passt. Es verfügt über eine automatische Polaritätserkennung und Temperaturerfassung (sowohl auf der Montageplatte als auch beim externen Kabel). Programmierbar sind nominelle Ladeströme von 400...5000 mA.

Auf der Montageplatte befinden sich verstellbare Federkraftkontakte, die auf beliebige Batteriekontakte zugreifen können. Das externe Ladekabel mit Krokodilklemmen (und einem Thermistor als Sensor) erreicht größere Batterien oder einen Apple-Adapter. ⏮

(160253)

https://goo.gl/ar15x7

Zusammengestellt von **Robert van der Zwan**

Fast Forward Award 2016: Wind und… Wasser

Zum Thema Wind wird heute sehr viel geforscht, doch wie sieht es beim Wasser aus? Viele Wasserzähler sind immer noch sehr ungenau. Die FFA-Jury fand den hochpräzisen Zähler des Australiers Lenn Williams so einzigartig, dass sie dafür sogar den neuen „Tech for Good“ Preis erfand und verlieh. Lenn (Foto, mit Brille) kehrte mit einem Scheck von 5.000 Euro in seine Heimat zurück, den er für PR und Marketing sowie für die Verbreitung seiner Erfindung verwenden darf.

Wir müssen uns nicht selber loben…

… denn das machen bereits andere für uns: Falk Senger, der Geschäftsführer der Messe München, betonte, dass er mehr als glücklich über den ausgezeichneten Ablauf des *Fast Forward Awards 2016* war und ihn die Art und Weise, wie die Vielfalt der gezeigten, qualitativ hochwertigen Innovationen einem breiten Publikum näher gebracht wurde, sehr beeindruckt hatte. Die von Elektor geleitete Jury des FFA hätte es nicht treffender formulieren können.

electronica 2016 in Zahlen

Blicken wir kurz auf die electronica 2016 zurück: Zahl der Aussteller: 2913 Firmen aus über 50 Ländern. Besucherzahl: 73.000 Besucher aus 80 Ländern (und damit 7 % mehr als im Vorjahr). Hauptthemen: Sicherheit und Automation. 99 % der Besucher beurteilten die Messe in München als „gut“ oder „ausgezeichnet“.

Die Projektleiterin der electronica Angela Marten und der Elektor-Geschäftsführer Don Akkermans unterschreiben hier eine Vereinbarung, dass die Zusammenarbeit beim electronica Fast Forward Start-up Award fortgesetzt wird. Der nächste FFA wird auf der electronica 2018 verliehen – doch schon nächstes Jahr geht es los mit den Vorbereitungen.

Wenn die Sonne Energie aus lauen

Die Endrunde des Fast Forward Awards begann mit großer Spannung an einem Dienstagmorgen um 10 Uhr am großen Elektor-Messestand auf der electronica. Unter dem Motto „Innovation“ wurden 35 Erfindungen aus 16 Ländern vorgestellt. Die Entscheidung, welche Einsendungen in den drei Kategorien Prototyp, Start-up und Idee zu den Gewinnern zählen sollten, fiel der internationalen Jury nicht leicht. Diese setzte sich übrigens aus Elektor-Mitarbeitern sowie aus Mitarbeitern renommierter Unternehmen wie STMicroelectronics, Würth Elektronik und Conrad zusammen.

Nach längerer Beratung gelangte die Jury schließlich zu einem Ergebnis: Mowea aus Berlin siegte im Bereich „Prototyp“, BotFactory aus New York belegte den ersten Platz in der Kategorie „Startup“ und Kewazo aus Garching war Gewinner in der Kategorie „Idee“. Am Freitag traten alle Sieger noch einmal gegeneinander an, sie mussten dabei noch zusätzliche Fragen der Jury und des Publikums beantworten.

Um 11.00 Uhr gab der Jury-Vorsitzende Clemens Valens zunächst den Gewinner des Preises in der Kategorie „Idee“ bekannt: Kewazo mit seinem Gerüstbau-Roboter. Artem Kuchukov wurde zum glücklichen Besitzer eines Schecks im Wert von 25.000 Euro für PR- und Marketing-Zwecke.

Im Rahmen der Kategorie „Start-up“ überreichte Clemens Valens dem Geschäftsführer J.F. Brandon von BotFactory anschließend einen Scheck von 50.000 Euro für einen mit einem kompletten Bestückungsautomaten ausgerüsteten Desktop-Printer.
Die Nummer eins wurde Mowea mit seinen Mikro-Windturbinen, mit denen sich auch in kleineren Haushalten und Unternehmen Wind- und Sonnenenergie zu einer netzunabhängigen,

PEOPLE NEWS • Tanja Pohlen sorgte, genauso wie Angela Marten von der electronica, für die reibungslose Organisation
Tessel Renzenbrink und Jens Nickel unsere Online-Leser über alles auf dem Laufenden, was sie über das Event wissen
Anchorman Jan Buiting und sein Kameramann Patrick Wielders auch die kamerascheuesten Aussteller zur Preisgabe
-Kollegen Julia Grotenrath, Chantalle Reuling und Raoul Morreau haben am Elektor-Stand zahlreiche Gespräche mit

mal nicht scheint: Lüftchen

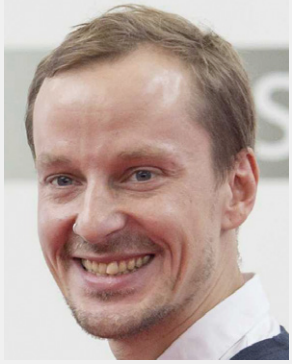
stabilen Stromversorgung kombinieren lassen. Geschäftsführer Till Naumann und sein Team empfingen einen Scheck über 75.000 Euro. Sie dürfen sich im Jahre 2018 auch noch über einen Stand auf der electronica freuen.

Foto: Dr. Till Naumann und Lara Obst von Mowea präsentieren ihre Wind-Turbine.

des Fast Forward Awards • Während der Messe hielten mussten • Vor Ort in München ermunterte unser Elektor-TV von Hintergrundinformationen • Die Elektor-Kolleginnen und Besuchern geführt. ... ...

EXPERTENPROFIL

Elektor arbeitet mit mehr als 1.000 Experten und Autoren bei der Produktion von Büchern, Artikeln, DVDs, Webinaren und Live-Events zusammen. In jeder Ausgabe wollen wir einen von ihnen ins Rampenlicht stellen...



Name: **Dr. Till Naumann**

Alter: **38**

Ausbildung: **PhD, Technische Universität Berlin**

Berufliche Interessen: **Ich liebe den Wind und die Aerodynamik, vielleicht, weil mein Vater Stunt-Pilot war.**

Wer ist Till Naumann?

Ich bin ein Ingenieur aus Berlin und habe dort an der TU meine Doktorarbeit zum Thema Aerodynamik und Windturbinen verfasst. Mit meinen Kenntnissen über große Windturbinen würde ich gerne bewirken, dass kleine Windturbinen für jedermann erschwinglich werden. Meinen Geschäftspartner Andreas [Amberger, die Redaktion] habe ich an der TU Berlin kennen gelernt. Zusammen entwickelten wir ein modulares Windenergie-System mit einer nominalen Leistung von 10 kW.

Waren Sie überrascht, den FFA der electronica zu gewinnen?

Unser Team war sehr überrascht. Wir kamen mit dem Zug aus Berlin und konzentrierten all unsere Energie auf die Präsentation unseres Prototypen. Nach fast drei Tagen ohne Schlaf sind wir nun sehr glücklich, dass wir es geschafft haben!

Haben Sie schon einen Plan, wie Sie die 75.000 Euro ausgeben, oder ist es dazu noch zu früh?

Als Start-Up-Unternehmen brauchen wir ein gutes, vielseitiges Marketing, ein Corporate Design, eine bessere Homepage, Demo-Videos unserer Prototypen, Inhalte für unsere B2B-Partner, Gelegenheiten zur Präsentation auf entsprechenden Messen und eine Community – zum Beispiel bei facebook oder anderen sozialen Medien. Voraussichtlich für das Ende des Jahres 2017 planen wir gerade eine Crowdfunding-Kampagne.

Was hoffen Sie, in den nächsten fünf Jahren zu erreichen?

Wir möchten ein solides aber dennoch schnell wachsendes Unternehmen etablieren, mit dem wir kostengünstige, saubere und netzunabhängige Lösungen im Energiebereich anbieten - und mit dem wir an der Dekarbonisierung der Energieerzeugung teilhaben können.

Was hat Sie zur Entwicklung Ihres Windenergie-Plans veranlasst?

Ich liebe Wind und Aerodynamik, weil mein Vater ein Stunt-Pilot war. Schon als Kind verbrachte ich sehr viel Zeit in der Luft. Bei Andreas und mir wurde die Leidenschaft für kleine Wind-Anlagen durch das weltweite, große Interesse an überschaubaren Lösungen im Bereich erneuerbarer Energien geweckt. Vor allem für Länder, denen es an der erforderlichen Infrastruktur mangelt.

Was wird Ihre wichtigste elektronische Entwicklung in nicht allzu ferner Zukunft sein?

Wir wollen unsere Turbinen „intelligent" machen, um damit Daten über die Energieproduktion und das Verhalten der Kunden zu gewinnen. Außerdem wollen wir unsere Turbinen in Mikro-Netze einbinden – es gibt also, um es auf den Punkt zu bringen, noch viel für uns zu tun.

Katzdoku

Sudoku für Elektroniker

Von **Claude Ghyselen** (Frankreich)

Zum Jahresbeginn haben wir unser traditionelles Hexadoku durch ein „Katzdoku“ ersetzt. Das entsprechende Gitter sieht etwas anders aus, wobei sich jedoch die Regeln für die erlaubten Zeichen und die Art, sie einzutragen, nicht geändert haben.

Zur Erinnerung: Ein Hexadoku-Rätsel setzt sich aus Hexadezimalzahlen zusammen (von 0 bis 9 und von A bis F). Jedes Zeichen darf dabei in jeder Reihe, jeder Spalte und in jedem Fach nur einmal erscheinen. Die Fächer sind durch eine fett gedruckte Umrandung gekennzeichnet.

Das Katzdoku unterscheidet sich vom Hexadoku durch folgende Merkmale:

- Das Gitter enthält 16 Fächer, aber 20 Zeilen (1 bis 20) und 20 Spalten (A bis T) (siehe unten).
- Fünf Zonen sind leer (weiß).
- Die Fächer sind nicht quadratisch oder rechteckig, sondern gleichen vielmehr einem Puzzle-Teil.
- Das Fach 1 (Pupillen in hellgelb) besteht aus zwei getrennten Teilen (linkes und rechtes Auge).
- Die Anzahl der Zellen eines Fachs hängt von dessen Form ab, überschreitet jedoch nicht die Zahl 16.
- Die Gesamtzahl der Felder beträgt 204, im Gegensatz zu 256 beim Hexadoku.

Da einige Fächer weniger als 16 Zellen besitzen, lassen sich dort nicht alle hexadezimalen Zahlen unterbringen - eine kleine, zusätzlich eingebaute Hürde auf dem Weg zur Lösung.

Die in den Feldern sichtbaren Ziffern legen die Anfangskonfiguration fest. Wenn Sie einen Gutschein gewinnen möchten, müssen Sie nicht das ganze Rätsel einsenden, sondern nur die 6 Ziffern in den grauen Feldern.

Wer das Rätsel löst - sprich die Ziffern in den grauen Kästchen herausfindet - kann einen von drei Gutscheinen im Wert von 50 Euro gewinnen!

Zu Fächern gruppierte Kästchen

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	**Anzahl der Kästchen pro Zeile**
1				3												8					2
2				3	3										8	8					4
3				3	3	3								8	8	8					6
4				3	3	3	3		6	6	7		7	8	8	8					11
5				3	3	3	3	6	6	6	7	7	7	8	8	8	8				14
6				3	3	6	6	6	6	6	7	7	7	8	8	8	9				14
7				4	4	4	6	6	6	6	7	7	7	9	9	9	9				14
8				4	4	4		1		6	7	7		1		9	9	9			11
9			2	4	4			1			10			1			9	9	9			9
10			2	4	4			1			10			1			9	9	11	11		10
11			2	4	4	4		1		10	10	10		1		9	9	9	11	11	14
12			2	5	5	5	10	10	10	10		10	10	10	10	10	11	11	11		16
13		2	2	5	5	5	15	15	12	12		12	12	13	13	11	11	11			16
14	2	2	2	5	5	5	15	15	12	12		12	12	13	13	11	11				16
15		2	2	5	5	15	15	15	12				12	14	14	14					12
16				5	5	15	15	15	12	12	12	12	12	14	14						12
17						15	15	15	15	15	14	14	14	14							9
18								16	16	16	16	16	16								6
19										16	16	16	16								4
20											16	16	16	16							4
n	1	3	7	16	15	13	10	14	11	13	13	13	13	16	11	13	10	6	4	2	**204**

Block-Nummer	Anzahl der Kästchen
1	8
2	11
3	16
4	13
5	13
6	15
7	13
8	16
9	16
10	14
11	12
12	15
13	4
14	9
15	15
16	14
	204

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1																9				
2				8											0	F				
3				B										7		1				
4				1	6		3			0	4			A	5					
5				D	5	9			B						4		2			
6				0				E		3	7		F	B						
7								6		F	8	B	3		9					
8				5	E	B					0			D			C			
9			2	6	D			A						4			E	B		
10			6	C	2			F						1					4	B
11			F			A						8		3		5	1		D	E
12								B	1						7	0	8	9	2	
13		0										1		E	C		A			
14	D		9			7				2							5			
15		4	3				B	0					7		D					
16				9	F		D				E	3	B							
17						F	9	1	3	6			2							
18									6		9	A	4							
19										8	5	7								
20											B	D	0							

Einsenden

Schicken Sie die Lösung (die Zahlen in den grauen Kästchen) per E-Mail, Fax oder Post an:

Elektor Redaktion
Kackertstr. 10
52072 Aachen

Fax: 0241 / 955 09-013
E-Mail: hexadoku@elektor.de

Als Betreff bitte nur die Ziffern der Lösung angeben!

Einsendeschluss ist der 31. Januar 2017.

Die Gewinner des Hexadokus aus der November-Ausgabe stehen fest!

Die richtige Lösung ist: **389BD.**

Einen Elektor-Wertgutschein über je 50 € haben gewonnen:

Johann Parfuss, Claude Viou und Bernd Lörler.

Herzlichen Glückwunsch!

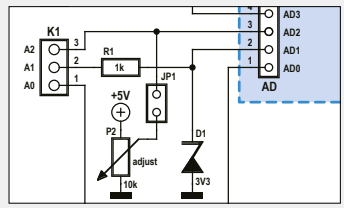
3	0	7	B	4	D	8	E	5	2	9	1	F	6	A	C
4	2	5	1	7	C	F	6	8	A	D	0	9	B	E	3
C	E	8	F	B	A	9	2	3	4	6	7	D	0	1	5
6	D	9	A	0	1	3	5	B	C	E	F	2	4	7	8
E	3	A	0	D	F	2	8	C	9	B	5	7	1	4	6
8	F	D	4	9	5	B	1	A	0	7	6	3	2	C	E
7	9	B	2	6	E	C	3	D	8	1	4	5	A	F	0
5	1	C	6	A	0	4	7	2	E	F	3	8	9	B	D
0	4	F	8	C	2	7	9	E	3	A	D	1	5	6	B
B	5	2	D	3	4	A	0	1	6	8	C	E	F	9	7
9	6	1	7	8	B	E	D	F	5	0	2	A	C	3	4
A	C	E	3	5	6	1	F	7	B	4	9	0	D	8	2
D	7	0	5	E	3	6	A	9	F	C	B	4	8	2	1
F	8	4	C	1	9	5	B	0	7	2	E	6	3	D	A
1	A	6	9	2	7	0	C	4	D	3	8	B	E	5	F
2	B	3	E	F	8	D	4	6	1	5	A	C	7	0	9

Der Rechtsweg ist ausgeschlossen.
Mitarbeiter der in der Unternehmensgruppe Elektor International Media B.V. zusammengeschlossenen Verlage und deren Angehörige sind von der Teilnahme ausgeschlossen.

IoT-Shield für den Arduino

Elektor 12/2016, S. 40 (160169)

Im zweiten Absatz des obengenannten Artikels muss es „Analogeingang A2“ heißen, nicht „Analogeingang A3“.



Projekt 2.0

Korrekturen, Updates, Leserbriefe

Kontaktlose Energieübertragung

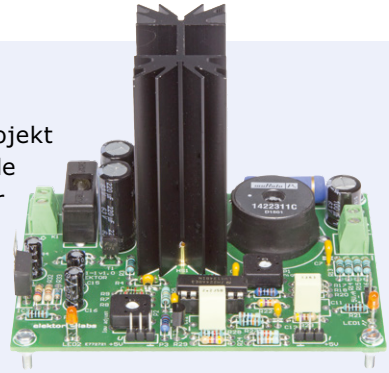
Elektor 11/2016, S. 26 (160119)

Auf Seite 29 wurde der Hinweis im roten Kasten versehentlich nicht übersetzt. Hier die deutsche Übersetzung:

„Achtung: Versuchen Sie niemals, ein Qi-Gerät (wie z.B. ein Smartphone) mit dieser Schaltung zu laden!“

UniLab 2

Elektor 10/2016, S.46 (150161)



Ich fand Ihren Artikel zum UniLab 2 sehr spannend und werde das Projekt wohl demnächst nachbauen. Die Beschaltung des ungenutzten OpAmps finde ich aber keine gute Methode. Hier ein Hinweis zum Anschließen ungenutzter OpAmps: https://e2e.ti.com/blogs_/archives/b/thesignal/archive/2012/11/27/the-unused-op-amp-what-to-do

Björn Schuster

Hallo Herr Schuster,
durch jedes bisschen Offset am Eingang wird der Ausgang entweder hoch oder niedrig. Wenn es ein Ultra-Low-Offset-Opamp wäre, käme er dem idealen Opamp sehr nahe und es könnte eventuell Probleme geben. Aber hier ist das nicht zu erwarten. Generell ist der obengenannte Hinweis aber sicher eine gute Empfehlung.

Ton Giesberts (Elektor-Labor)

Elektor SDR Reloaded (4)

Elektor 11/2016, S. 42 (160165)

In der Erklärung zu Bild 5 bis 7 schreibt Burkhard Kainka, dass der 1-nF-Kondensator bei 16 kHz mit dem 10-kΩ-Widerstand einen 90-Grad-Phasenschieber bildet. Ein einfacher RC-Tief- oder Hochpass hat aber bei der -3-dB-Frequenz nur 45 Grad. Für eine korrekte Funktion müsste man mindestens zwei hintereinanderschalten.

Zur DSP-Verarbeitung der IQ-Signale, wie sie im letzten Absatz des Artikels angesprochen wird, habe ich mal vor ein paar Jahren einen Wiki-Artikel verfasst:
www.mikrocontroller.net/articles/Hilbert-Transformator_(Phasenschieber)_mit_ATmega
Ein Arduino könnte durchaus ausreichen, um einen digitalen 90-Grad-Breitbandphasenschieber zu berechnen. Leider hat mein erster Versuch geschwungen, was bei IIR-Filtern und einer 16-bit-Auflösung keine Seltenheit ist.

Christoph Kessler

Hallo Herr Kessler,
es handelt sich in diesem Fall um kein Tiefpassfilter, weil am invertierenden Eingang des Opamps eine virtuelle Masse liegt. Die Schaltung ist ein Addierer, der die Eingangsströme zu einer Ausgangsspannung macht. Für die Ströme gilt aber über einen weiten Frequenzbereich der Phasenunterschied 90 Grad.
Danke für den Hinweis zum Hilbert-Transformator! Da muss ich allerdings gestehen, dass ich hier an meine Grenzen komme. Aber wenn es nicht Leute geben würde, die sowas für alle nutzbar umsetzen können, gäbe es auch keine SDR-Software.

Burkhard Kainka

enova STRASBOURG

THE TECHNOLOGIES **EXHIBITION** FOR TOMORROW'S **INNOVATIONS**

ELECTRONICS / EMBEDDED / IoT / MEASUREMENT / VISION / OPTICS / BIG DATA

MAKE YOUR PROJECTS A REALITY

AERONAUTICS I MILITARY I AGRIBUSINESS I AGRICULTURAL I AUTOMOTIVE I TRANSPORT
SMART CITIES I SMART BUILDING I MEDICAL I INDUSTRY 4.0 I ACADEMIC RESEARCH

Made by GL events

www.enova-event.com

reichelt elektronik
Technik verbindet.

Jetzt Newsletter abonnieren!
Unter allen neuen Newsletter-Abonnenten wird monatlich ein **technisches Highlight** verlost!
http://**rch.lt/v3**

Foto: Foto- und Bilderwerk

„Mit den Microcontrollern von Arduino werden Ideen ganz einfach Wirklichkeit!"

Der Beliebteste unter den Arduinos!
ARDUINO UNO REV. 3
- ATMega 328
- 14 digitale Ein- und Ausgänge (davon 6 als PWM)
- 6 analoge Eingänge
- USB-Anschluss

ARDUINO UNO **19,50**

ARDUINO

Klein, aber nicht zu unterschätzen!
ARDUINO™ MICRO
- ATMega 32u4
- 20 digitale Ein- und Ausgänge (davon 7 als PWM)
- 12 analoge Eingänge
- micro-USB-Anschluss

ARDUINO MICRO **18,50**

Einfach anschließen und loslegen!
ARDUINO™ MEGA 2560
- ATMega 2560
- 54 digitale Ein- und Ausgänge (davon 14 als PWM)
- 16 analoge Eingänge
- USB-Anschluss

ALLE ARDUINO PRODUKTE
rch.lt/yb

ARDUINO MEGA **34,90**

Klein, aber oho!
ARDUINO™ NANO
- ATMega 328
- 14 digitale Ein- und Ausgänge (davon 6 als PWM)
- 8 analoge Eingänge
- mini-USB-Anschluss

ARDUINO NANO **22,90**

Tagespreise · Preisstand: 01.12.2016, Preise in € inkl. gesetzl. MwSt., zzgl. Versandkosten, reichelt elektronik, Elektronikring 1, 26452 Sande (D)

- über 45 Jahre Erfahrung
- schneller 24-Std.-Versand
- über 70.000 Produkte am Lager
- kein Mindermengenzuschlag
- zertifiziert nach DIN EN ISO 9001:2008

www.reichelt.de
BESTELLHOTLINE: +49 (0)4422 955-333